ON CONTROL OF MOTION PATH AND SPEED IN A SPLINE-BASED ANIMATION

BY

WOUSEOK JOU

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1991

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor and supervisory committee chairman, Dr John Staudhammer, for his invaluable inspiration, advice and help Without his guidance, this work could never have been done.

I am greatly indebted to Dr Paul W Chun, Department of Biochemistry and Molecular Biology, for the encouragement and financial support to start working on this dissertation. His stringent instruction and progressive guidance in biochemical issues made this work possible Also, this work was supported by a faculty development grant, Division of Sponsored Research at University of Florida

I would like to thank sincerely all the members of this committee. Professor Jack R. Smith and Professor Jose C Principe of the Department of Electrical Engineering, Professor Hong H. Lee and Professor Chang W Park of the Department of Chemical Engineering, for their recommendation and advice regarding this dissertation.

Also, I wish to express my gratitude to Dr Richard Feldmann of the National Institutes of Health for offering me a motivation for the application part of this dissertation Dr Arun Varma, Professor of Mathematics, helped me find relevant materials for this research Thanks are also due Randy Bowling, who arranged the use of a Silicon Graphics 4D/210 GTX workstation for energy minimization This dissertation also benefitted from the discussions I had with Dr Roy Vaz of Tripos Inc, developers of Sybyl software

Finally, Thanks are due my wife Chung-ae for the care of our home and our children through the years of the preparation of this dissertation.

TABLE OF CONTENTS

page

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

ON CONTROL OF MOTION PATH AND SPEED IN A SPLINE-BASED ANIMATION

By

WOUSEOK JOU

August, 1991

Chairman: Dr. John Staudhammer
Major Department: Electrical Engineering

Various graphical models for the visualization of pharmacologically-active proteins have been developed to help recognize a three-dimensional structure of molecules, these models have been principally aimed at visualizing the static images of molecules. This work is an extension of such computer representations to animated display of molecular activities such as the folding of protein molecules. This dissertation describes a new graphical model suitable for the animation of protein folding.

During animation, the motion path of individual atoms forming a molecule may exhibit an undesirable wiggle or inflection under certain conditions. The G^1 class of splines can be used to eliminate these artifacts, but a mathematical representation of the G^1 splines has not been developed to

date. We develop an *analytic expression* for the G^1 splines. Moreover, we extend and apply the G^1 splines for the generation of more diversified motion path in an interactive animation environment.

This dissertation develops a method termed the *incremental knot spacing* to control the motion speed by an approximation in the discrete parameter domain so that control of motion speed is achieved. We separate temporal aspect from spatial aspect of animation while providing a means of hitting key frames.

Based on these major tools for the control of motion path and speed, we develop a molecular animation model termed the *solid backbone model*, in which a molecular backbone is depicted as a combination of shaded cylindrical surfaces. Most importantly, the inbetweening technique was incorporated into the model to smooth the motion transition.

CHAPTER 1
INTRODUCTION

Protein Folding

In late 1957, Christian B. Anfinsen, a researcher at the National Institutes of Health, made a remarkable discovery while experimenting with the early notions of protein folding that postulate the existence of "templates" that cause proteins to assume their native structure [Voe90]. What causes newly made proteins—which resemble loosely coiled springs—to wind into a specific shape?

His experiment confirmed that a native molecule of ribonuclease can fold into a denatured structure and back into the exact original native structure [Ros81]. Under proper conditions, the denatured shape remembers its original native shape and completely recovers it. Therefore, it was generally believed that the amino acid sequence of a protein, a one-dimensional trait, was fully sufficient to specify the molecule's ultimate three-dimensional shape and biological activity.

The next question to arise was "How does the protein fold between the two states?" For more than 30 years after Anfinsen's discovery, hundreds of investigators examined this problem [Cho84, Wri88] in an attempt to predict the folding pathway. The solution is of more than academic concern. The major impetus behind the biotechnology industry is to produce new and useful proteins. It is already possible to design genes which could guide the synthesis of such proteins. Yet an essential feature of such synthesized proteins must be that they be able to attain the natural, stable state with full biological activity by undergoing folding. If, however, the synthesized proteins do not fold as intended, the resulting proteins cannot exist in vivo.

Because only a split second is required for folding (e.g., 1-100 ms) and because the methods that provide structural information about protein folding (e.g., X-ray crystallography, Nuclear Magnetic Resonance) are intrinsically slow [Kim82], no one has been able to witness the folding process. Instead, biochemists have managed to trap a few folding intermediates in a stable form by slowing down the process. These folding intermediates constitute the basic building blocks for the various models which have been proposed to illustrate the folding process [Kim82]. However, there is as yet no known solution to the problem of prediction of the folding pathway [Ghé82, Ric91].

With the advent of the computer age, theoretical endeavors could complement earlier experimental work. For example, the shape of a folded protein might in principle be determined by an empirical surface-potential-energy function. Given the numerical values of the charges of atoms, the resulting energy may be expressed as the sum of pair interactions between all atoms of the protein [Ghé82, Syb89]. Factors that influence the potential-energy function (e.g., van der Waals forces, bond stretching energy, side chain torsional energy) depend heavily on the postulated model [Bro83, Cla89, Lev83a, Syb89].

Based on these factors, the computer adjusts the coordinates of the atoms so that the overall energy is lowered and a minimum is reached. The minimum energy state is the natural, stable state of the proteins. This process, known as energy minimization, has been a valuable tool not only for confirming models of structures but also for determining folding pathways by simulation [Jou91, Chu91, Lev83b, Rac91] Although some problems in this approach have been reported [Ghé82], progress in this area of research has been rapid. This dissertation, however, does not deal with the modeling and computational processes involved in such energy minimization

Besides their use as a minimization tool, protein engineers have found computers useful for the display of protein structures. Interactive computer graphics put bond structures, molecular volumes, and surface areas under the control of protein engineers with a simple interface devices [Piq86] The protein images can be rotated, magnified and represented as suits the user Fragments of proteins can be manipulated to approach one another and joined to test the ease of fit of a substrate and a binding site. Clues from the visualization of the binding contribute to the design of substrates that bind more effectively and enhance the activity of complex systems [Mil91] In essence, computer graphics plays a key role in protein engineering

Various graphical models [Lev66, Max83, Sch79] for the representation of three-dimensional structures of proteins have been used to aid in the recognition of proteins—real or simulated

The skeletal model is based on the Kendrew-Watson physical model of a molecule [Sch79] and shows the molecular framework which is a collection of lines joining atoms The skeletal model, often referred to as wireframe model [Str81], implies atomic position by the junctions of bonds and their termini. Although the skeletal model has capability of rapid imaging of large molecules, recognition is restricted to intramolecular topology such as bond lengths bond angles, and relative atomic positions.

The spherical model illustrates molecular surface shape as formed by the union of van der Waals contact surfaces Van der Waals distance, which is used as atomic radius in the spherical model, is defined as the distance where a repulsive force begins to appear when two atoms are drawn together. Although the spherical model enables enhanced perception and conceptualization of critical interactions [Chu90] commonly occurring at this contact surface, existing graphics hardware does not permit interaction with a spherical model larger than a few tens of atoms [Sta78, Fal88].

The ribbon model [Car86, Ric81] visualizes the backbone of proteins, defined as the repetition of nitrogen, alpha-carbon, and carbonyl-carbon atoms. Common implementation of this model represents the backbone line as a ribbon composed of multiple parallel, smooth threads running along its length.

With the growing interest in protein folding, the obvious extension of such computer representation is an animated graphic display of the motion of a molecule. Feldmann and Levitt [Fel80], in a 1980 film of the molecular dynamics of bovine pancreatic trypsin inhibitor (BPTI), demonstrated the power of computer animation as a tool for the researcher. From a collection of moving pictures, all aspects of the reaction pathways could be seen directly. Fundamental concepts were easy to grasp using visual effects involving motion. The techniques of computer animation show promise in applications ranging from non-invasive robotic surgery to the prediction of structure for pharmacologically-active enzymes and the animated display of such molecular processes as DNA unwinding or protein folding process.

Nevertheless, these existing molecular models are appropriate for the representation of the static view of protein structures. In applying them for molecular animation, however, a number of serious problems arise. The flipping of side chains in the skeletal model diverts the viewer's attention. The spherical model, besides being prohibitively expensive means to produce images for individual folding intermediates, hides the backbone with the outside surfaces of atomic spheres. This dissertation deals with the definition of a new model which avoids these inherent flaws and is specifically designed for use in molecular animation

Statement of the Problem

Considerations in a Static View

Energy minimization is a predominant simulation technique to predetermine a reaction pathway, since large molecular movements which involve concerted displacements of many atoms or those requiring the crossing of energy barriers cannot be investigated with molecular dynamics alone [Fuj86, Lev83b] Because protein folding is associated with a globular conformational change, molecular structures resulting from energy minimization have been widely used in modeling the folding intermediates [Del84] Our animation system also uses energy minimization as a tool for the generation of folding intermediates.

A peptide plane in a protein is defined as the rigid plane containing carbon, oxygen, nitrogen and hydrogen atoms which reside between two adjacent alpha-carbons belonging to corresponding amino acids Therefore, a protein molecule may be regarded as a sequence of the peptide planes joined by alpha-carbon atoms

Because the bond lengths between atoms are fixed, only the angles joining each of the peptide planes can freely rotate Consider the rotation of the peptide planes induced by a movement of a *single* alpha-carbon atom A local rotation of a peptide plane propagates through the entire chain length to relieve the constraints imposed on the bond lengths From a mechanical viewpoint, this situation is somewhat similar to the rotation of the arms with respect to the body in a multi-arm robot system [Fu87] When only the joint between body and first arm is allowed to rotate, while the rest of the arms have no degree of freedom to rotate, all of the arms will undergo exactly the same degree of rotation as the first arm

Energy minimization, however, does not calculate the angular rotation of the peptide planes in a one-by-one manner Rather, energy minimization adjusts the coordinates of a whole group of localized atoms in such a way that the overall energy is lowered and a minimum is reached, without concern for the individual rotations of the peptide planes Thus what we see as a result of energy minimization is the location of atoms which have gone through combinations of rotations centered at several joint positions

The wireframe model has been used most frequently in previous attempts at molecular animation because of the high computation cost required when the spherical model is used In this model, the atomic positions are represented by the junction of bonds and their termini Nevertheless, the wireframe model exhibits the following fundamental flaws as an animation tool

As a result of the combination of rotations propagated from several joint positions, the angles joining the bonds can vary randomly What makes the animation possible is the visual illusion of the first image remaining until the second image is displayed If the next image conspicuously deviates from the previous one, and deviation continues through the entire succession of frames, conceptualization of movement cannot occur Since the angles joining the bonds do not assure gradual changes, the viewers cannot rely on them as a source of visual illusion. The bonds linking the junctions suffer the same loss of visual continuity in the wireframe model Since the junctions of bonds move at apparently random way, the lines connecting these junctions cannot show continuous movement. Figure 1-1 shows an animation sequence of part of a hemoglobin molecule. Numbers in this figure designate the frame sequence numbers. As can be noticed, it is hard to find a reference object which is moving in a visually smooth manner, in this example of the wireframe model. Instead, such disturbing features as flipping side chains and abrupt changes in bond angles can take away the viewer's attention during animation.

Figure 1-2 illustrates a ubiquitin molecule—whose abnormal structural deformation is known to be characteristic of Alzheimer's disease—shown in the spherical model. Besides the enormous computation involved in rendering the individual spheres, the backbone atoms are buried under an outside overcast of spherical surfaces.

What viewers tend to conceptualize in the animation of folding is the global structural change represented by the backbones of proteins. This can be evidenced by numerous illustrations [Mer77, Str81, Voe90] where the folding process is drawn by thick and smooth backbone lines. Defining backbone lines as the combination of the lines joining the sequence of nitrogen, alpha carbon and carbonyl-carbon atoms, biochemists prefer to interpret the folding process in terms of the extension and contraction of these backbone lines.

Characterization of the folding process is possible with the backbone lines. Because amino acids stick to each other to form proteins we call the individual amino acid the monomer, or primary structure. Newly-formed proteins are often called random coils, implying that no region of backbone looks significantly different from other regions. Certain portions of these coils, which are probably unstable and fluctuating, may serve as "seeds," around which stable coils will eventually stick. The backbone of proteins thus formed has some peculiarities. For example, the usual backbone of the native molecule can be divided into regions of secondary structure, which are distinct segments having a characteristic conformation. They may form helices, flat sheets, or turns connecting the helices and strands of sheets. The presence of different secondary elements raises the possibility that certain amino acids tend to be found in a specific secondary structure [Sch79]. For instance, some amino acids are found more often in helices than in random coils. Backbone lines in this context play an important role in conceptualization of secondary structure.

An additional important advantage of using backbone lines to represent the molecule is that the backbone lines can completely recover geometrical information of entire atoms. Protein designers quite often require the ability to stop the animation flow and query about the geometrical information between atoms in a static view of a scene. Naturally they may use a mouse to click on a portion of the backbone atoms which draws their attention during animation. Because the two peptide planes centered at alpha-carbons are rigid units, clicking on the position of alpha-carbons can generate coordinates of other atoms on those rigid units. Provided with a proper data structure to link the alpha-carbons with the rigid units, these atoms can be viewed in the spherical model. Therefore, it is possible to view the globular changes in animation and to focus on and analyze the spherical shapes in a static scene. The bridge between the two modes is an exact click on the location of the alpha-carbons.

The ribbon model has succeeded in helping the user visualize the backbone lines. This model, however, has some inherent problems. First, conventional implementations of the ribbon model do not pass through the backbone atoms. In Figure 1-3, the central thread of the ribbon is the primary curve approximating the backbone line, and the remaining threads are constructed to be parallel to it However, it is not assured that the central thread will hit the position of the alpha carbon, a major constituent of the backbone atoms. Most of the visual implementations [Car86, Syb89] estimated the smoothness property so highly that approximating splines became dominant in representing the backbone lines In part, popularity of approximating splines stems from the fact that the interpolating splines show wiggles in markedly-twisted helices The approximating splines, to be described later in this chapter, are not guaranteed to pass through the sample points which in this case are the backbone atoms. Since the threads of the ribbon model do not necessarily reveal the location of the backbone atoms, the position of the backbone atoms cannot be marked on the backbone curve, and the viewer loses the bridge between the animation and static view

Second, in the ribbon model, it is difficult to recognize depth since the only clue to the three-dimensional understanding of backbone lines is the twist and width change of parallel lines of that ribbon model Often, the threads are intersecting and the decision as to which backbone conformation is closer is left to the viewer's experience. The viewer needs time to examine such details to form a good understanding of the spatial nature of the backbone In animation, however, cine rate limits the duration of a static frame. The animation stream cannot arbitrarily assign a long time to figure out the three-dimensional structure depicted by multiple ribbon threads Even if this model is used to make individual frames of the animation sequence, the multiple threads become another source of flipping, as were the bonds in the wireframe model.

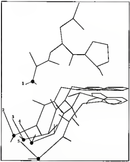

Figure 1-1. A schematic diagram of the animation of a wireframe model Black dots represent positions of an identical atom in different frames

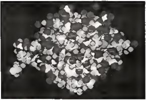

Figure 1-2. Spherical model of a ubiquitin molecule

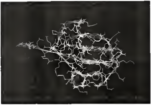

Figure 1-3. Ribbon model of the ubiquitin molecule in Figure 1-2. Wireframe model is overlapped in the background

Considerations in a Dynamic View

A general misconception of previous animation systems in molecular graphics [Del84, Fuj86, Tod83, Syb89] is that animation may be performed by simply displaying the individual frames consecutively. If the individual frames are of a relatively similar spatial pattern, this assumption may be true. However, folding intermediates, whether they are determined from a theoretical postulation or they are taken from experimental results including a computer simulation such as energy minimization, tend to be dissimilar. Therefore, the resulting frames exhibit abrupt and unnatural displacement from one frame to the next. Such abruptness can be noticed in Figure 1-1 where the transition between Frame 1 and Frame 2 is conspicuous when compared with other transitions.

Key frames derive their name from the key drawings [Tha85] on the celluloid transparencies used in conventional animation. If we define the individual folding intermediates of the energy minimization as the key frame, we can employ the traditional computer graphics technique called inbetweening. Inbetweening [Bur71, Ree81, Tha85] works as follows: the animator specifies two key drawings and the computer calculates and generates additional drawings between them by referencing the distance between pairs of corresponding points. When applied to the display of folding intermediates, the technique will generate additional imaginary intermediates between the two folding intermediates. Movement from one folding intermediate to another can be improved by the introduction of additional inbetween frames. These inbetween frames smooth the abrupt conformational discontinuities between the folding intermediates. Since what we are concerned with is the visualization of the global movements of a backbone structure, introduction of these inbetween frames does not interfere as long as it aids in the capturing of motion.

Suppose an atom has to travel from point A to point B as a result of energy minimization such that point A belongs to the first folding intermediate and point B belongs to the next one. There can be various motion paths from A to B depending on the inbetweening scheme. The path trajectories might be straight lines or smooth curves. Splines, in a broad sense, can be said to be the smooth curves connecting points A and B. If the curves pass through the two points, the splines are said to be interpolating. In contrast if the curves approach and do not pass the two points, the splines are said to be approximating. For the situation at hand, the atom does not hit the target destination if approximating splines are used. This means that the approximating splines used as inbetweening tools will modify the structure of the original folding intermediates, making entire frames imaginary ones. If a user stops the animation stream to analyze the static structure, he has no place to anchor because the original key frames do not belong to the animation sequence. Therefore, switching between animation and static analysis becomes difficult with the approximating interpolations.

In an attempt to pass through the key frames, interpolating splines may replace the approximating splines. The motion path passes through the exact locations of the original folding intermediates used as key drawings. This spline, however, introduces the problems of wiggles, loops, or oscillation. Suppose that the atom proceeds to move from point B to another point C. If the distance from A to B is large compared with the distance between the points B and C (as in the black dots of frame 1, 2, and 3 of Figure 1-1), the motion path tends to wiggle in the interval [B, C]. We would like to avoid this misbehavior of the motion path in our animation system.

In addition to the problem related to the motion path, there remains another problem associated with the control of motion speed. If the atom accelerates and decelerates unnecessarily, transition between folding intermediates will become fluctuating as well. Such motion speed is attained from sampling the points lying on the motion path. With more points sampled, more inbetween frames are produced and the motion will be slower. The major factor determining the motion speed is the spatial distance between sample points. Traditionally, it has been known as a difficult problem to adjust the motion speed on a curve produced by a parametric representation. Because the splines used in computer graphics are predominantly expressed parametrically, the inter-frame distances should also be controlled by the parameter. However, the relation between the parameter and the inter-frame distance is not linear in general, and this nonlinearity complicates the control of the motion speed.

Objectives and Approach

We can restate the problems as follows:

Problem 1. The spherical model is not suitable for folding animation since the backbone conformation may be invisible.

Problem 2. The wireframe model is not suitable for the animation because of its inherent discontinuities in spatial pattern.

Problem 3. The ribbon model cannot be used directly for animation because the viewer needs time to examine the twist and width of the parallel lines

Problem 4 Since the threads of the ribbon model do not pass through the backbone atoms generally, information about atomic locations is lost.

Problem 5 Backbone curves drawn with interpolating splines may show wiggles in some regions.

Problem 6 Folding intermediates tend to be dissimilar.

Problem 7 The technique of inbetweening by approximating splines can miss the key frames

Problem 8. Inbetweening by interpolating splines can exhibit an abnormal motion path because of the wiggles

Problem 9. Even if interpolating splines are used, motion speed may not be controlled

This dissertation resolves Problems 1 through 3 by devising a new model called the solid backbone model. The solid backbone model shares the basic idea of embodying backbone lines with the ribbon model The backbone can sufficiently convey all the information about folding while reducing the time constraints of the spherical model. However, the solid backbone model replaces the multiple threads of the ribbon model with a series of shaded cylinders to speed up the visual recognition process during animation

Being defined as a series of cylinders with their axes on the backbone lines, the solid backbone model gives the feeling of three-dimensional depth by proper shading of the cylindrical surfaces It is a known fact that, in animation, relatively rough shading will suffice, compared with static scenes Since the trend in current hardware is to facilitate rapid computation, generation of rough shading of three-dimensional objects is a simple process For instance, most current graphic devices are equipped with VLSI circuits dedicated to shading. Rendering features related to the solid backbone model are presented in Chapter 5.

Problem 6 can be handled by applying the inbetweening technique. Such inbetweening is a "must" to enable the recognition of the smooth transition of folding pathways. Were it not for the role of inbetweened frames, the entire motion sequence would be a discrete array of spatially irrelevant images. The inbetween frames help the viewer conceptualize the motion.

Problem 4 and problem 7, although apparently different, are practically the same in light of the interpolation schemes. If the approximating splines are applied to atomic positions in space to produce backbone curves on a static scene, problem 4 occurs. If the same approximating splines are applied to the motion path of a single atom to produce the path trajectories over a span of time, problem 7 occurs. Since visual interactivity is extremely important in current applications of computer graphics, the transition between the animation mode and the static mode should be secured right from the beginning. Therefore this dissertation adopts interpolating splines both for the determination of static shapes of individual frames and the determination of the motion path of individual atoms between the frames. This way, the backbone curves will pass through the backbone atoms and the motion path will hit the key frames made up of folding intermediates.

The adoption of interpolating splines naturally invokes problem 5 and problem 8, which are inherent misbehaviors of interpolating splines in space and time domains, respectively. Could we remove the wiggles from the interpolating splines? Chapter 3 resolves these problems by developing a new class of interpolating splines termed free form splines.

Problem 9 is unavoidable whether we choose approximating splines or interpolating splines. Could the motion speed be controlled at will? Chapter 4 establishes an algorithm to control the motion speed on the path trajectories produced by spline techniques in general

In summary, the aims of this dissertation can be categorized as follows:

1 Development of wiggle-free interpolating splines.

2. Control of the motion speed in animation.

3 Development of a new model for molecular animation.

Chapter 3 deals with the first category while Chapter 4 handles the second Chapter 5 describes the development of a prototype system as an implementation of the model in the third category

Organization

Chapter 2 explains fundamental concepts involved in the spline techniques used in computer graphics. Spline functions in the context of parametric curves are stated Visual continuity, one of the most important concepts used in developing our new class of splines, is described in this chapter

Chapter 3 develops a new class of interpolating splines called free form splines, which are visually continuous. Chapter 3 is organized as follows.

1 Motivation in relation to molecular animation is presented.

2. A survey of the previous research associated with the removal of wiggles is made

3. Mathematical definition of general free form splines is presented.

4. Mathematical definition of visually continuous free form splines is presented

5 Four characteristic types of free form splines are defined in terms of relevant tangent vectors.

6. Performance of the four types of splines is measured on the removal of the wiggles.

7 A comparison of the free form splines and the conventional approach is made.

8 As an application of the free form splines, procedures for an interactive design of motion path, are described

Chapter 4 develops a method for controlling the motion speed in animation It is organized as follows

1 The motivation for control of the motion speed is described in terms of molecular animation.

2 Research relevant to the control of speed is summarized.

3 As a framework of our control method, a method called dynamic knot spacing is defined.

4 A pair of enhancement methods are presented as a tool for further refinements of the dynamic knot spacing

5 The dynamic knot spacing and the enhancement methods are represented by a pseudo-code

6. Error analysis of the algorithms is presented.

Chapter 5 describes an application of the above techniques to the scripts of protein folding. This chapter develops a prototype system that uses the free form splines developed in Chapter 3, both for the rendering of the backbone and the motion path. The dynamic knot spacing developed in Chapter 4 is incorporated into the prototype system to control the motion speed in animation Chapter 5 is organized as follows

1. Following an introduction, general background knowledge of protein chemistry as applicable to the problem is summarized

2 A literature review of molecular graphics in general, and molecular animation in particular, is made.

3. Fundamental concepts delineating our animation model are described

4 Considerations involved in the design of the prototype system are explained

5. Experimental frames resulting from the prototype system are illustrated

Chapter 6 summarizes the major accomplishments and contributions of this dissertation

CHAPTER 2
PREREQUISITES AND REVIEW OF SPLINE FUNCTIONS

The exploration of the use of parametric curves and surfaces can be viewed as the origin of Computer Aided Geometric Design (CAGD). Until the 1960's the only reliable tool for the communication between the designers and the manufacturers had been numbers. For instance, in a car-body engineering, the surface had to be totally expressed with numbers. Since the numbers were at best a discrete approximation of the continuous real shape, a set of curves was carved in a 3D model and actual interpolation to mold the shape was left to the experience of highly skilled patternmakers.

With the introduction of the Bézier curve [Béz70], CAGD had a major breakthrough. Being parametrically represented, this curve scheme enables all data to be expressed exclusively by a few numbers. Since the specification of numbers defines the entire curve shape, the shape could be standardized regardless of the experience of the patternmakers. Taking advantage of this approach, mathematics became involved in generating a variety of curve schemes that could interpolate or approximate given numbers so that CAGD could be further applied to the design process in general.

This chapter describes the prerequisite terms and definitions for such interpolation, and presents previous work done in this area of computer graphics to facilitate further discussion in subsequent chapters. Only those terms necessary for the explanation of our approach will be given and the coverage of previous work will be limited to this purpose. Research done in very specific areas will be addressed in later chapters.

Often, a purely mathematical definition of a term involving redundant symbols and terminologies makes a concept extremely difficult for graphic designers to understand. Therefore, preference is given to more concise yet precise representation of such a mathematical definition. For example, the matrix representation of a spline function is easier to understand [Smi83] and thus has been more widely used than polynomial representation. The descriptions in this chapter are given in such a way that complicated mathematical definitions are minimized.

Parametric Curves

Although a curve may be presented by a collection of points, provided they are closely spaced, there are several reasons why a mathematical representation is popularly used in computer graphics. The advantage of a mathematical representation is that it is precise and the properties of curves such as slope can be easily calculated from it. Moreover, it can be stored compactly in a computer in the form of equations.

Either a parametric or a nonparametric form can be used to represent a curve mathematically. An explicit nonparametric space curve is given by a set of equations of the form

$x = x$

$y = f(x)$

$z = g(x)$

where

f and g are arbitrary mappings from x to y and z, respectively

For instance, let f (x) be

$ax^2 + bx + c$

so thet

$$y - (ax^2 + bx + c) = 0$$

where

a, b, c are some arbitrarily coefficients to be solved

Given the coordinates of three data points in space, it is simple to get the a, b, and c value so that the x-y relationship of the curve passing through the points can be calculated However, such nonparametric expressions are inadequate for the purpose of computer graphics. The reasons are as follows

1 Frequently, the situation arises where coordinates of one data point and the slopes of two data points are specified instead of specifying three coordinates If the slope is taken from a data point where the curve slope is perpendicular to the x-axis, the value of the slope will become infinity, making evaluation of the coefficients impossible.

2. If the curve shows closed forms such as loops, there will be multiple values of y corresponding to a single x value, and the evaluation of the coefficients becomes difficult.

3 When points on a nonparametric curve are calculated with equal increments in x, the position of the point will not be distributed evenly along the length of the curve so that the quality and accuracy of the graphical output is affected

These difficulties are caused by axis-dependency, and can be avoided by using parametric representation commonly used in computer graphics In parametric form, each coordinate of a point on a curve is represented as a function of one or more parameters For a curve with one parameter, the position vector for a point on a curve is expressed by the parameter For instance, the three Cartesian components of a space curve in terms of a parameter t, are written as

$$x = f_1(t)$$
$$y = f_2(t)$$
$$z = f_3(t)$$

where

f_1, f_2, f_3 are arbitrary functions from t to x, y, z, respectively

Since a point on a parametric curve is specified by a single parameter value, the parametric form is axis-independent. For instance, the equation of a unit circle in its nonparametric form

$$x = x$$
$$y = \pm\sqrt{1-x^2}$$

can be converted into parametric form as

$$P(\theta) = (x, y)$$

with

$$x = \cos\theta$$
$$y = \sin\theta.$$

While angle θ varies from zero to 360 degrees, a closed circle can be generated without producing the dual y values appearing in the nonparametric representation The tangent vector of a point on the circle, with respect to the parameter θ, is given by

$$P(\theta) = (-\sin\theta, \cos\theta)$$

In contrast to the infinite tangent of the nonparametric form with an x value of one, the parametric tangent with a θ value of zero does exist and is not infinite

These advantages of parametric representation have led to its popularity as a tool for computer graphics, so thet most interpolation schemes are developed with parametric notation Our approaches appearing in the subsequent chapters will also be developed and explained with parametric representations

Spline Functions

The interpolation theory of mathematics has borrowed the term spline from its physical counterpart, a mechanical device used by draftsmen to fit a curve of minimum curvature through successive points of a set. Physically, a spline [Hea86, Rog76, Smi83] is a flexible strip which is bent under the weight of heavy metals placed to control the resulting shape By varying the number and position of the lead weights, the spline can be made to pass through the specified data points such that the resulting curve appears smooth or fair

From a mathematical point of view, the problem of defining a curve from a known set of data points is one of *interpolation*. For instance, a curve can be made to pass through all known points by use of polynomial interpolation. It is important to note thet a clear distinction is made between the definition of the above interpolatory polynomial and the polynomial used as a spline For instance, given ten data points, an *interpolatory polynomial* is a single polynomial passing through all ten points On the other hand, a *spline* is a piecewise polynomial passing through a subset of the ten data points The complete curve is produced by linking the adjacent polynomials at some joints between the subsets In the example, three of the splines, each passing through four consecutive data points, can be combined to form the final curve In general, the mathematical spline is a piecewise polynomial of degree K with continuity of derivatives of order $(K-1)$ at the common joints between segments. For instance, the cubic spline has second order continuity at the joints.

In practice, the function representing a spline does not have to be polynomial. It can be any function including the exponential and trigonometric functions. However, the polynomial gained popularity in computer graphics because it is simple and easy to evaluate, and infinitely differentiable on those points of a curve which do not belong to data points.

A curve generated by such pieces of splines, expressed in terms of the parametric representation, can be defined as a continuous map of a collection of intervals $[u_1 < \ldots < u_n]$ into three-dimensional space. Figure 2-1 illustrates the mapping from the knot variable *u* into the parameter *s* and the mapping from s into the space curve *P(s)*. Initially, points $\mathbf{P}_1$ through $\mathbf{P}_5$ can be regarded as the points on the curve **P**(u) corresponding to the parameter u_1 through u_5 so that the curve **P**(u) is the mapping from the u-parameter domain into the three-dimensional space. Each real number u_i in this sense is defined as a *breakpoint* or a *knot* and the collection of all u_i is called the *knot sequence*. Note that the formal definition of the knots distinguishes knots from data points. The data points are the points in three-dimensional space while the knots are the corresponding parameter values used for the generation of the space curve that interpolates them. A local parameter s for the interval $[u_3, u_4]$ is defined by setting

$$s = \frac{u - u_3}{u_4 - u_3}$$

so that the *normalized* parameter s can range from zero to one while the global parameter u varies from u_3 to u_4. It is by this mapping from the u-parameter domain into the s parameter domain that every piecewise spline can be expressed in terms of the normalized parameter $(0 \le s \le 1)$. Because of this mapping, one can indiscriminately generate the expression for each piecewise polynomial without paying attention to the actual values of the knot parameter u. Subsequently, curve $\mathbf{P}(s)$ is a collection of piecewise polynomials as a function of the parameter s.

Since the polynomial is infinitely differentiable on the curve, the problem of differentiability is concerned with the data points making up the joints between two adjacent piecewise splines. For instance, suppose a cubic spline $\mathbf{P}_1(s)$ that interpolates two data points $\mathbf{P}_1, \mathbf{P}_2$ with normalized parameter s. Furthermore, assume that we know the derivatives at the points, $\mathbf{P}_1$ and $\mathbf{P}_2$. Then the equation of a cubic polynomial curve satisfying these four constraints is

$$\mathbf{P}_1(s) = [s^3 \; s^2 \; s \; 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix}. \tag{2.1}$$

This equation is for one cubic spline segment and can be generalized for any two adjacent cubic segments. If additional cubic spline segments, $\mathbf{P}_2(s)$ through $\mathbf{P}_{n-1}(s)$ are assumed to interpolate intervals $[\mathbf{P}_2, \mathbf{P}_3]$ through $[\mathbf{P}_{n-1}, \mathbf{P}_n]$, the curve interpolating the points $\mathbf{P}_1$ through $\mathbf{P}_n$ is made up of a combination of piecewise splines $\mathbf{P}_1(s)$ through $\mathbf{P}_{n-1}(s)$.

Let us apply the preceding definition of a mathematical cubic spline which is two times differentiable (C^2 *continuous*) at the joints. To impose the second order continuity constraint at the joints, we are supposed to specify the positions of the n data points and the two endpoint derivatives, $\mathbf{P}_1$ and $\mathbf{P}_n$. The first order derivatives at intermediate data points cannot be specified arbitrarily. Instead, they are calculated by the constraints of second order continuity For instance, if the number of data points n is equal to three, the derivative at the point $\mathbf{P}_2$ is calculated by setting

$$\left.\frac{d^2\mathbf{P}_1(s)}{ds^2}\right|_{s=1} = \left.\frac{d^2\mathbf{P}_2(s)}{ds^2}\right|_{s=0}$$

In general, the first order derivatives of the intermediate data points are computed by solving the tridiagonal matrix equation

$$\begin{bmatrix} 4 & 1 & 0 & 0 & & & \\ 1 & 4 & 1 & 0 & 0 & \ldots & \\ 0 & 1 & 4 & 1 & 0 & \ldots & \\ & 0 & 1 & 4 & 1 & 0 & \\ & & & & & & \\ & & & & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} \mathbf{P}_2 \\ \mathbf{P}_3 \\ \vdots \\ \vdots \\ \vdots \\ \mathbf{P}_{n-1} \end{bmatrix} = \begin{bmatrix} 3(\mathbf{P}_3\text{-}\mathbf{P}_1)\text{-}\mathbf{P}_1 \\ 3(\mathbf{P}_4\text{-}\mathbf{P}_2) \\ 3(\mathbf{P}_5\text{-}\mathbf{P}_3) \\ \vdots \\ \vdots \\ 3(\mathbf{P}_n\text{-}\mathbf{P}_{n-2})\text{-}\mathbf{P}_n \end{bmatrix} \qquad (2.2)$$

where

n is the the number of data points to be interpolated,

$\mathbf{P}_1$ and $\mathbf{P}_n$ are the first and last data point.

The coefficient matrix of this equation is diagonally dominant, the solution exists and is unique [Far90]. The calculated first order derivatives are used to produce the corresponding piecewise splines in each interval.

Although the above mathematical cubic spline is twice differentiable at data points, it is not *local* and therefore, it is seldom used in computer graphics. That is, if a position of a point is modified, the above system of linear equations should be evaluated again for all the first order derivatives. Consequently, the change of a single data point propagates and modifies entire piecewise splines. This loss of locality has kept the mathematical cubic spline from being widely used, since an attempt to modify only a portion of a curve will affect all other portions of a curve which are already completely designed. In contrast, if the cubic spline has only the continuity of first order derivatives, the interpolating curve changes only in the vicinity of that point. The tradeoff for the lower differentiability is locality of the splines. Our research is mainly concerned with those kinds of cubic splines which preserve locality.

The most common splines employed in computer graphics applications are local cubic splines, local because displacement of a knot affects curve shapes only in the vicinity of that point, cubic because third order is appropriate to get relatively smooth interpolations with reasonable computational complexity. Moreover, it is of the lowest degree space curve which allows a point of inflection and thus has the ability to twist through space [Rog76]. Local cubic splines can be categorized into *interpolating splines* which pass through their data points and *approximating splines* in which data points are used only to control the curve shapes. To provide more smoothness near data points, the approximating splines have continuity of the second order derivative. The price paid is that the data points, with the exception of endpoints, are no longer on the curve. The data points act on the shape which they typically define, only remotely. Because of this property of the approximating splines, the data points are often called control points.

Because of the undesirable feature that the data points are not on the curve, approximating splines require practice in figuring out the resultant shapes when the sample points are changed. In general usage, the data points are repetitively modified on a trial and error basis until an acceptable shape is achieved. Because of this property, approximating splines are often used as a modeling tool for rough approximation of the curve and final curve fit is done using interpolating splines.

Interpolating splines are referred to by several different names: the *cardinal spline*, the *Catmull-Rom spline*, or the *Overhauser spline*. Tensioned Catmull-Rom splines are often called cardinal splines. Interpolating splines pass through given data points and this property makes them ideal for our animation system despite their relative shortcoming that sample points have only the first order derivative continuity

Visual Continuity Conditions

This section describes a visually continuous class of splines which will be mathematically defined, proved, and extended in the next chapter

Smoothness of a curve [Rog76, Lan86] is defined in terms of its differentiability from a mathematical point of view. A curve which is two times differentiable is smoother than one with the continuity of first order derivative *near data points*. As is known from fundamental calculus, the term *differentiable* or to have *continuous derivative*, means that both the left and right derivatives exist and that their magnitudes and directions are identical. To fulfill this requirement, individual derivatives of the x, y, z component of Cartesian coordinates with respect to a parameter should be of the same magnitude

Nevertheless, it is a known fact in computer graphics that the parametric tangents do not provide an appropriate measure of continuity in the context of geometry [Bart87, Bars88, Far90]. For instance, Figure 2-2 shows two piecewise first order splines given by

$$\mathbf{P}_1(s) = [2s\ ,\ s] \qquad (0 \le s \le 1)$$

$$\mathbf{P}_2(s) = \left[\frac{1}{2}(s+2)\ ,\ \frac{1}{4}(s+4)\right] \qquad (0 \le s \le 1)$$

and the parametric first derivative vectors are

$$\mathbf{P}_1(s) = [2, 1]$$
$$\mathbf{P}_2(s) = \left[\frac{1}{2}, \frac{1}{4}\right]$$

Even if the segments join with a discontinuous parametric first derivative vector, the joint at [2 , 1] is visually continuous. Figure 2-3 is another example illustrating the fact that the visual continuity does not necessarily coincide with the continuity of the parametric derivative. The line segments are,

$$\mathbf{P}_1(s) = [a(2s - s^2) , b(2s - s^2)] \qquad (0 \le s \le 1)$$
$$\mathbf{P}_2(s) = [a + (c - a)s^2 , b(1 - s^2)] \qquad (0 \le s \le 1)$$

Therefore, the parametric first derivative vectors become

$$\mathbf{P}_1(s) = [a(2 - 2s) , b(2 - 2s)] \qquad (0 \le s \le 1)$$
$$\mathbf{P}_2(s) = [2(c - a)s , -2bs] \qquad (0 \le s \le 1)$$

Since both derivatives evaluated at the joint are the same with the value of zero, the parametric first derivative is continuous at the joint positioned at (a b) Mathematically, this continuity is called the C^1 *continuity* Nevertheless, the two line segments at that point exhibit a cusp which is far from visually continuous

Visual continuity is defined in terms of the continuity of the unit tangents If the unit tangent of the left and right segment at a joint is identical, it is visually G^1 *continuous* The unit tangents of the line segments at the joint are the same in the Figure 2-2,

$$\mathbf{P}_1(s) = \mathbf{P}_2(s) = \left[\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}}\right]$$

so that it is G^1 continuous. On the other hand, those of Figure 2-3 are

$$\mathbf{P}_1(s) = \left[\frac{a}{\sqrt{a^2 + b^2}}, \frac{b}{\sqrt{a^2 + b^2}}\right] \qquad (0 \le s \le 1)$$
$$\mathbf{P}_2(s) = \left[\frac{(c - a)}{\sqrt{b^2 + (c - a)^2}}, \frac{b}{\sqrt{b^2 + (c - a)^2}}\right] \qquad (0 \le s \le 1)$$

Except for the c value of 2a, it is not G^1 continuous by definition.

By defining visual continuity as the continuity of unit tangent vectors rather than the tangent vectors, the mathematical continuity condition could conform to the visual perception In addition to maintaining the same visual smoothness as the parametrically continuous splines, the class of the G^1 continuous splines encompasses a wider range of splines than C^1 continuous splines These advantages of the visually continuous splines will be extensively exploited in Chapter 3.

Literature Review

In approximation theory, mathematicians have developed various numerical analogs of splines, well before cubic splines were widely used in the computer graphics field Among approximating functions, polynomial interpolation became a focus for study because polynomials are easy to evaluate, differentiate and integrate, and are well-behaved As a result, classical (precomputer) numerical analysis was concerned with finding a polynomial which, when interpolated between tabulated values of functions, could best recover the original function values

The Lagrange form of interpolating polynomial [Jac70, Jac81] expressed in terms of parameter s is given by

$$\mathbf{P}(s) = \sum_{i=0}^{n} \mathbf{P}_i L_i^n(s)$$

with the Lagrange polynomial

$$L_i^n(s) = \frac{\prod_{j=0\,(j\neq i)}^{n}(s - s_j)}{\prod_{j=0\,(j\neq i)}^{n}(s_i - s_j)}$$

where

P_i ($i = 0..n$) are data points to be interpolated,

s_j ($j = 0..n$) are knot values corresponding to P_i ($i = 0..n$)

Despite its concise formulation, Lagrange interpolation has two problems which limit its applicability in computer graphics First, it oscillates as the order of the polynomial increases and the shapes of the original curves are not guaranteed to be preserved This effect is often referred to as the "Runge phenomenon" [Far90] Second, modification of one point affects the whole curve shape at the expense of costly computation. That is, the Lagrange polynomial lacks in locality In most cases, this is not desirable for design purpose since modification in one portion of a curve must not jeopardize other completed portions by changing the entire shape

The Bézier curve [Béz70, Béz74, For72] represents one of the earliest attempts to develop a flexible curve scheme in computer-aided design. Starting with de Casteljau's recursive algorithm [Bart87, Far90], Bézier derived an equation for a piecewise approximating polynomial, which was used by Renault to design car bodies. As adapted by Bernstein [For72, Gor74a], the closed form of the Bézier curve is

$$B^n(s) = \sum_{i=0}^{n} B_i \, B_i^n(s)$$

with the Bernstein polynomial

$$B_i^n(s) = \binom{n}{i} s^i \, (1-s)^{n-i}$$

where

n is the order of the Bézier curve,

s is the normalized knot parameter,

B_i ($i = 0..n$) are data points to be interpolated.

Notice that the curve passes through the two end points B_0, B_n but not the internal points. One of the important properties of the Bézier curve is the *convex hull* property [Far90]. Since the coefficients B_i^n (s) are all nonnegative in addition to summing to one, they form a *convex combination*. A convex combination of points is always inside those points. This observation leads to the definition of the *convex hull* of a point set as the set that is formed by all convex combinations of a point set. More intuitively, the convex hull can be said to be a polygon bounded by the data points in the planar case.

A Bézier curve of the above form represents a smooth curve over a range of control points. These piecewise Bézier curves can be joined end-to-end to form a composite Bézier curve. At the joints of the individual curve segments, certain continuity conditions need to be met to preserve the overall smoothness of the composite curve. Maintaining a C^1 or higher continuity [Car76] in a composite Bézier curve requires that the change of control points satisfy certain constraints [Far90] on knot ratios. Therefore, if some control point is modified arbitrarily to reshape curves adjacent to that point, the previous continuity condition of nearby curve segments may be destroyed.

The B-splines [Gor74b] were devised such that the same order of continuity remains after modification of control points. Geometric interpretation of B-splines in terms of the Bézier curve can be found in [Far90, Bart87]. Sharing the *basis* function with the Bézier curve, the B-splines inherit the convex-hull property [Har87] from the Bézier curve. Matrix representation [Smi83] of the cubic B-spline is

$$P(s) = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix}$$

where

s is the parameter value in the range of $(0 \le s \le 1)$,

$\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ are internal control points corresponding to $s = 0$ and $s = 1$,

$\mathbf{P}_{i-1}$ and $\mathbf{P}_{i+2}$ are control points neighboring $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ respectively

Noticing that continuity of parametric derivative is not a proper measure of visual continuity, Barsky's β spline [Bars83, Bars88] incorporated a G^1 continuous [Kari87] tangent vector with first and second order continuity. It is important to note that the β spline belongs to the class of approximating splines. Hence the G^1 continuity is preserved at some points on the curve corresponding to the data points, but not at the data points themselves. Expressed explicitly, continuity of the unit tangent vector and the curvature vector persists if

$$Q_2^{(1)}(0) = \beta_1 Q_1^{(1)}(1)$$

$$Q_2^{(2)}(0) = \beta_1{}^2 Q_1^{(2)}(1) + \beta_2 Q_1^{(1)}(1)$$

where

$Q_2^{(1)}(0)$ is the 1st order left derivative at vertex,

$Q_1^{(1)}(1)$ is the 1st order right derivative at vertex,

$Q_2^{(2)}(0)$ is the 2nd order left derivative at vertex,

$Q_1^{(2)}(1)$ is the 2nd order right derivative at vertex, and β_1, β_2 are shape control parameters

With the simple observation that a curve can preserve its visual continuity when the unit tangent vector rather than the tangent vector is identical for the left and right derivatives, Barsky could naturally deduce beta parameters as a by-product. Upon inspection of these two parameters (β_1, β_2) for their effect on the shape of a curve, they are referred to as *bias* and *tension* parameters, respectively The bias parameter controls the direction of the slope at a data point, while the tension parameter controls the linearity of the curve segments near data points

The cubic B-spline and the β spline are powerful modeling tools, they are able to model complex shapes easily This "modeling" is carried out as an approximation process, manipulating the control points until a desired shape is achieved However, the interpolating process should also be incorporated even if one uses approximating splines as a modeling tool. It takes the role of final refinement The precision requirement that the curve pass through a given data set can now be imposed on the rough curve generated by the approximation process. While both B-spline and β spline have discarded interpolatory character to get the elegance of C^2 continuity, there have been andeavors to deduce a class of interpolating splines which is capable of generating a smooth interpolating function with C^1 continuity only

Catmull [Cat74] crafted the "Catmull-Rom spline" by separating blending function [Har87] from the cardinal functions being blended. This concept applies to any blending function and to any cardinal function to produce a wide class of splines. As one example of this class of splines, blending linear cardinal function with a B-spline basis function yields a cubic interpolating polynomial This polynomial, in matrix notation, becomes

$$\mathbf{P}(s) = [s^3\ s^2\ s\ 1]\ \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix}$$

where

$\mathbf{P}_k$ $(k = (i-1) \ldots (i+2))$ are the control points,

s is the parameter value in the range $(0 \le s \le 1)$.

As was intended, this cubic polynomial maintains interpolatory character at the expense of C^2 continuity.

The above special case of Catmull-Rom spline was not first to appear. Brewer and Anderson [Bre77] repeated and described a lesser known spline called the Overhauser curve [Ove68] which was developed well before the Catmull-Rom spline. The Overhauser method, developed at Ford Motor Company in 1968, resorted to parametric blending function without any a priori knowledge of splitting or of B-splines. Given four consecutive data points, $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4$, a parabola **A** can be made to interpolate the points $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$. Similarly another parabola **B** can interpolate $\mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4$. Then the linear blending of the parabola **A** and **B** yields the cubic spline which is effective on the region $[\mathbf{P}_2, \mathbf{P}_3]$. This method forms the platform for the development of the new class of splines discussed in Chapter 3.

Often designers need to specify the slopes instead of interpolation points. In the case of the cubic spline, for example, they may indicate positions and slopes at two internal points instead of positions at four points. The question is, "Given data points and corresponding tangent vectors, find a C^1 piecewise cubic polynomial thet interpolates the given data."

Cubic Hermite interpolation [Jac87, Far90], when applied to the end point derivative conditions of Bézier curves, takes the following Bézier form.

$$\begin{aligned} \mathbf{P}(s) &= \mathbf{P}_i B_0^3(s) + \mathbf{P}_{i+1} B_3^3(s) \\ &\quad + (\mathbf{P}_i + \frac{\mathbf{D}_0}{3}) B_1^3(s) + (\mathbf{P}_{i+1} - \frac{\mathbf{D}_1}{3}) B_2^3(s) \end{aligned}$$

where

$\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ are two end points of the region of concern,

$\mathbf{D}_i$ and $\mathbf{D}_{i+1}$ are slopes corresponding to $\mathbf{P}_0$ and $\mathbf{P}_1$,

$B_i^3(s)$ $(i = 0..3)$ is the Bernstein polynomial on parameter s

In matrix form, the above equation simplifies to Equation 2.1 with $\mathbf{P}_i$ replaced by $\mathbf{D}_i$ Interestingly, this formulation of the Hermite polynomial includes the Catmull-Rom spline as a proper set Catmull-Rom spline falls into the special case of Hermite polynomial if

$\mathbf{D}_i = (\mathbf{P}_{i+1} - \mathbf{P}_{i-1}) / 2$ and

$\mathbf{D}_{i+1} = (\mathbf{P}_{i+2} - \mathbf{P}_i) / 2.$

Kochanek [Koc84] noted this similarity and inserted multiplication constants in the expression for $\mathbf{D}_i$ and looked into the resulting changes in the curve shapes. Variation in these multiplication constants led him to define such parameters as bias, tension and continuity This method is compared in detail with our strategy for the control of curve shapes in Chapter 3

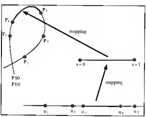

Figure 2-1. Mappings from u parameter into s parameter, and from s parameter into the piecewise spline curve P(s). The normalized parameter s varies between zero and one while the curve runs from P_3 to P_4

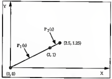

Figure 2-2 Discontinuous parametric derivative yet visually continuous case

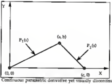

Figure 2-3 Continuous parametric derivative yet visually discontinuous case

CHAPTER 3
THE FREE FORM SPLINES

Introduction

The development of computer-aided geometric design relies on a wide range of mathematical methods for curve and surface fitting. The need for fitting curves and surfaces arises principally from the fact that many physical phenomena are continuous, although our measurement of them is discrete. From the discrete information, we try to reconstruct the continuum using the mathematical tool called spline interpolation.

In our situation, the position of atoms as represented by numbers constitutes such discrete information. Consider an atom out of a large molecule. During the folding process, the atom will change its position as a function of time to stabilize itself as dictated by the folding dynamics. However, because of the difficulties involved in the measurement, only a limited number of intermediate scenes can be captured experimentally. Therefore, the distance between the intermediate scenes will become unavoidably long.

The same holds true if such a folding process is simulated by a computer technique called energy minimization. For each step of the computation, the method tries to find a more stable atomic position by minimizing the sum of inter-atomic potentials, and the output of each step makes up a specific folding intermediate. As long as the new position yields less potential, the technique will reposition the given atom. Although the intermediate scenes can be more continuous if the distance is kept short, no specific concern is given about the distance an atom travels for each step of computation

There are two aspects in the animation of protein folding that require interpolation First, motion paths of individual atoms which make up folding intermediates should be interpolated. If an atom at position A at time t_1 moves into position B at time t_2, interpolation is needed to produce extra animation frames of the atomic positions between the time interval [A,B] Second, in a static scene at a fixed time, interpolation can be used to generate a backbone curve which passes through the positions of the backbone atoms. Because our animation system aims at viewing the folding process in the secondary structure level represented by a backbone curve, it is imperative to interpolate a given sequence of atoms Therefore, an interpolation scheme could be used both for the generation of a motion path of an atom and the generation of a backbone curve in a static scene

For reasons to be explained in Chapter 5, our animation model is based on the two principles in view of the interpolation scheme First, the curve representing the motion path should *pass through* the folding intermediates. In terms of the above example, the generated curve should hit both positions A and B Second, the curve representing the backbone should also pass through all the backbone atoms Therefore, usage of the *interpolating* spline rather than the *approximating* spline is essential for our animation system

The ordinary cubic cardinal spline a traditional interpolating spline which has been widely used, has an intrinsic undesirable visual property known as a wiggle Figure 3-1 shows the motion path of a backbone atom generated by a cardinal spline. In the time interval $[t_3,t_4]$, the motion path abnormally deviates from a reasonable track Often, this type of curve behavior is called the wiggle Due to the popularity of cardinal splines, there are numerous expressions describing similar behaviors such as, an extraneous *inflection*, an *oscillation*, an *overshoot*, a *kink*, or a *vibration* of the curve In addition to this critical demerit, the motion path of the interval $[t_2,t_3]$ is nearly linear, thus making the movement of the atom quite unnatural While the curve shows a wiggle in a relatively short interval, it tends to be linear in the longer interval. The linearity diminishes one of the visual advantages of cubic splines, that they can induce arbitrary inflection and twist so that the curve looks more smooth and natural. Although cardinal splines do not necessarily exhibit this property at all times, the inherent tendency to wiggle in short intervals and to linearize in the longer interval is always present and it is amplified as the distance between the intervals varies widely The same phenomenon can happen in the generation of the backbone curves. If the five atomic positions in Figure 3-1 are considered as five different atoms in the space domain on a given time, the backbone curve representing these atoms will suffer a similar problem, showing a loop and a line segment. This problem must be resolved for our system to work properly since the atomic positions during the folding process are not known.

This chapter is mainly concerned with the removal of the wiggles appearing in the ordinary cardinal splines. An example of our approach is illustrated in Figure 3-2. The wiggle during the time interval $[t_3,t_4]$ is removed by an interpolation scheme called the type I free form spline Moreover, the linearity in the interval $[t_2,t_3]$ can be alleviated by this scheme Most importantly, the tendency to wiggle and linearize becomes controllable, since it is based on a class of splines with G^1 continuity rather than the C^1 splines, to which the ordinary cardinal splines belong

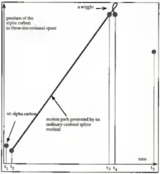

Figure 3-1 Interpolation by an ordinary cardinal spline. A curve representing the motion path of an alpha carbon is shown with a thick line. The alpha carbon is a constituent of the backbone atoms. Each t_i indicates the elapsed time during the folding process.

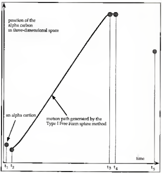
position of the
alpha carbon
in three-dimensional space
an alpha carbon
motion path generated by the
Type I Free Form spline method
time
t_1
t_2
t_3
t_4
t_5

Figure 3-2. Interpolation by Type I spline showing the removal of the wiggle and alleviation of the linearity. This class of splines assures the G^1 continuity at internal joints.

Literature Review

Wiggles can be regarded as an expression of tension inherent to the splines. If the wiggles are proparly controlled, they will lead to a visually pleasing inflection and twist. For instance, if the tendency to wiggle in Figure 3-1 is distributed to the longer interval, and the tendency to linearize is routed to the shorter interval, there remains no problem with the curve scheme. The control of tension has been investigated in this sense.

An approach to relieve the inclination to wiggle by the adjustment of tension can be found numerous times in the literature. The most prevalent one, the tensioned cardinal spline [Smi83], incorporates a tension parameter into the expression for the ordinary cardinal spline. The tensioned cardinal spline can straighten out the wiggles, or the extraneous inflection, by applying high tension in that region. However, while the wiggles in the shorter interval are straightened out by this method, the curve in a relatively longer interval tends to be linear as well. The fundamental flaw of this method is that the tension value cannot be applied for each piecewise interval because of the C^1 constraint.

Schweikert's cubic spline [Sch66, deB78] is called the spline under tension. Being C^2 continuous, this curve enables piecewise application of the tension parameter so that in each interval, the relative distance can interactively be reflected into the corresponding tension parameter. The only objection to this method is that it employs exponantial functions instead of polynomials, thereby making evaluation of the interpolant more expensive and in some cases, even imprecise. Nielson's v-spline [Far85, Far90, Nie74] is a polynomial alternative to the Schweikert's exponential function but unfortunately, the method lacks the mathematical proof of the existence of a unique solution [Bars83, Far90] to the linear system of equations that leads to the cubic spline.

Foley [Fol86 Sal84] developed a C^1 continuous cubic spline with the capability of piecewise tension control. Given the tensional weights in each interval, the method finds a cubic polynomial that minimizes the sum of weighted measure of curvature, subject to the continuity condition. Hence if a weight is set to be high in an interval, the curve will be assigned a low curvature so that it will be linearized Since each interval can be assigned a different parameter a piecewise control of the curve is possible. However, a change of weight in one interval propagates through the entire interval in this method The linear system of equations subject to the minimization constraint should be reevaluated even with a change of the position of a single data point Therefore, thus method lacks local controllability

The G^1 continuity encompasses a wider range of curve shapes than the C^1 class, and contains the C^1 class as a proper subset The principal reason why the G^1 class splines contain more diversified curve shapes is that the magnitude of the left and right tangents at a joint is allowed to be different Usually, the ratio of the left tangent and the right tangent takes the role of tension in this class of splines Since the magnitude of the tangents directly affects the tension [Man74], the tension of the curve segments to the left and to the right of the joint can independently be controlled in the G^1 splines.

In a sense, previous endeavors to control the tension while maintaining C^1 continuity at joints are unduly strained A consequence of the C^1 constraint can be explained by the split definition of the term tension It either represents making a curve tighter or looser *near* data points or *between* data points. Subject to a C^1 constraint, the curve tensioned near deta points [Nie74, Koc84] cannot ensure the tension between data points By the same token, the curve tensioned between data points [Bars84, Fol86, Sch66] cannot guarantee the similar degree of tension near the data points In contrast, in the G^1 splines, the tension of the curve segments near the data points and between the data points can be controlled independently by allowing the magnitudes of the left and right tangents to vary.

The properties of the G^1 continuity, the visual continuity, is investigated in [Man74] and the application of this type of continuity to the approximating spline appears in [Bars83, Bart87] The only endeavor to produce a G^1 cardinal spline is in [DeR88] The fact that the the basis functions of the cardinal spline (more generally, a Catmull Rom spline [Cat74]) can be expressed in terms of a Lagrange polynomials [Jac81, Lan86] is exploited in this approach Subsequently, the Bézier control vertices are formed geometrically (through an algorithm) for the piecewise Lagrange curves Although the existence of such control vertices is given without proof in the approach, by using such algorithms as recursive subdivision [Bart87, Lan80] or de Casteljau s algorithm [Far90], these control vertices then lead to the Bézier curve, which in this case corresponds to the G^1 cardinal spline Nevertheless, this approach is purely algorithmic, "owing to the algebraic complexity" [DeR88] involved

The explicit closed form of the G^1 cardinal spline could not be produced by this approach. Generally, a disadvantage of such algorithm-oriented approach is that a sufficient knowledge on the algorithm is required for its implementation As long as a graphics programmer could not understand it, it belongs to a complicated algorithm and it remains away from common usage Moreover, the most fundamental advantage of using the G^1 splines the control of tension by separate adjustment of the left and right tangents, cannot be fully exploited in this approach because no specific concern was given to the importance of tangent vectors

The G^1 interpolating spline explained in this chapter will be derived in explicit closed form starting with simple linear blending of a pair of points Since the spline presented in this chapter is represented by a single unique matrix, evaluation of the curve becomes a simple matter Furthermore, the tension parameters in this approach are closely related to the tangents at data points, so that piecewise tension values can be intuitively adjusted

Mathematical Description of the General Free Form Splines

Local interpolating splines have several characteristics which make them useful for the design process Because the curve follows and passes through the control points, it is possible to intuit directly where to put a new point Despite this convenience, previous uses of the cardinal splines have been limited because of the lack of proper tension control mechanism.

The fundamental flaw that causes the wiggles lies in the derivation of the cardinal splines. According to our observation, which will be shown shortly, the knots have direct consequences on the shape of a curve. Because mathematicians traditionally felt relatively comfortable in assuming equal knot spacing conditions, parametric splines which borrowed critical ideas from mathematics naturally followed this protocol For instance, Overhauser [Brew77, Ove68] assumed that the knot value of a midpoint is the average of two adjacent knots A similar assumption appears in Catmull-Rom spline [Cat74] by forcing symmetry on the cardinal function.

A cubic cardinal spline is a linear blending of two quadratics. Each quadratic can be represented as the linear blending of two consecutive line segments where each line segment, in turn, is a linear blending of two points.

Let us start with the linear blending of two points. Given points P_1 and P_2 in space, the linear blending of the two points is simply the line segment joining P_1 and P_2. Points on this line can be represented in a parametrized form

$$f(a) = (1-a)P_1+aP_2$$

where

a is a parameter in the range $(0 \le a \le 1)$,

f (a) is the positional vector of a point on the line.

For instance, point P_1 is the value of the function f evaluated at zero while point P_2 is the value of the function f evaluated at one. Therefore, as the parameter a runs from zero to one, the value of the function f makes up the continuous line segment from P_1 to P_2.

In Figure 3-3, f (a) and g (b) are two parametric functions which linearly interpolate points P_1, P_2, and P_2, P_3 respectively. The knot parameter a runs from zero to one making up the line segment P_1P_2, while the knot parameter *b* runs from zero to one making up the line segment P_2P_3. Therefore,

$$P_1 = f(0)$$

$$P_2 = f(1) = g(0)$$

$$P_3 = g(1)$$

Note that the two knot parameters are independent of each other.

We wish to blend the two line segments linearly so that resulting curve becomes a quadratic parabola passing through the three data points. Let us denote the parametric representation of this parabola as m(s). In order to normalize [Far90] the parameter s, we let $\mathbf{P}_1$ be the the function $\mathbf{m}$ evaluated at zero and let $\mathbf{P}_3$ be the function $\mathbf{m}$ evaluated at one:

$$\mathbf{P}_1 = \mathbf{m}(0)$$

$$\mathbf{P}_3 = \mathbf{m}(1)$$

The question is, "what is the knot value corresponding to midpoint $\mathbf{P}_2$?" Since point $\mathbf{P}_2$ is located somewhere on the curve joining $\mathbf{P}_1$ and $\mathbf{P}_3$, the corresponding knot value should be between zero and one However, the exact knot value is not known, and there is a degree of freedom in the assignment of the knot value corresponding to the midpoint $\mathbf{P}_2$ Because the quadratics at hand will eventually be blended to produce a cubic spline, and because the curve shape of the cubic spline heavily depends on the functions being blended, the knot value of the midpoint must be determined.

Before going further, we shall digress briefly to investigate the way ordinary cardinal splines are derived After an observation is made regarding the relation between the knot value and the resulting curve shape, our own derivation will be resumed This way, the concepts involved in the derivation of the free form splines can be clarified with ease

Ordinary cardinal splines regard $\mathbf{P}_2$ as a point mapped from the center of the parameter s domain such that

$$a = 1 \text{ at } s = \frac{1}{2}$$

$$b = 0 \text{ at } s = \frac{1}{2}$$

In order for this assumption to be valid, the linear relationship among three parameters should be

$$a = 2s$$

$$b = 2s\text{-}1$$

Hence, in the ordinary cardinal splines, the quadratic parabola produced by the linear blending of the two line segments is

$$\begin{aligned} \mathbf{m}(s) &= (1\text{-}s)\, f\,(a) + s\, g\,(b) \\ &= (1\text{-}s)\,((1\text{-}a)\mathbf{P}_1 + a\mathbf{P}_2) + s\,((1\text{-}b)\mathbf{P}_2 + b\mathbf{P}_3) \\ &= (1\text{-}s)\,((1\text{-}2s)\mathbf{P}_1 \text{-} 2s\mathbf{P}_2) + s\,((1\text{-}(2s\text{-}1))\mathbf{P}_2 \text{-} (2s\text{-}1)\mathbf{P}_3) \\ &= [s^2\ s\ 1] \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} \end{aligned}$$

Let us examine the change in the parabola by varying the knot value Our observation can be explained by Figure 3-4 Figure 3-4 shows the consequence of assigning two different knot values to the midpoint $\mathbf{P}_2$ during the formulation of the blending If the knot value corresponding to point $\mathbf{P}_2$ is set to .5, as is the case with ordinary cardinal splines, the parameter s domain is divided in the ratio of one to one The resulting curve $\mathbf{m}_2(s)$ exhibits a high inflection in the interval $[\mathbf{P}_1,\mathbf{P}_2]$, while it becomes relatively linearized in the interval $[\mathbf{P}_2,\mathbf{P}_3]$ However, with the division ratio of 1 4, the corresponding curve $\mathbf{m}_1(s)$ shows an explicit trend to counterbalance such differences in the degree of inflection. The curve shapes depend heavily upon the midpoint knot values, and we may infer the following

1 Suppose we are bending a wire with two hands The wire will bend more sharply if the distance between the two hands gets closer, provided that the wire has fixed length. An extreme case of this bending is the wiggle of the wire On the contrary, as the two hands are moving apart, the shape of the wire will tend to be linear The linearity of a curve is proportional to the distance between sample points

2. Consider two drivers assigned four and two hours respectively, for delivering a package from Gainesville to Jacksonville. The driver assigned four hours must spend more time on the road than the one who is assigned two hours. Thus, the route of the former driver will tend to meander, while the route of the latter driver tends to be linear. In the extreme case, the former driver may form a loop between the two cities. If Gainesville and Jacksonville are two points to be interpolated, the knots act as the time constraints assigned to travel the distance between the two cities. That is, the linearity of a curve is inversely proportional to the assigned knot values during the blending.

In order to offset the role of the knots as time constraints, we can make the knot values proportional to the distance between the sample points. That is, we allocate more time to travel to a greater distance.

In this section, however, we incorporate the knot values of midpoints as *parameters* of the quadratics and will delay the determination of the values until a cubic spline is produced by linearly blending the quadratics. It is in the cubic cardinal splines that we let the parameter values change to look into their visual properties. By allowing the knot values to vary, our approach can be explicitly contrasted with the ordinary cardinal splines.

Let us resume the derivation of our free form splines by referring to Figure 3-3. To make the knot value incorporated as a new parameter, we introduce a parameter u that indicates the knot value corresponding to the midpoint $\mathbf{P}_2$ (we will call this parameter the *midpoint knot parameter* or *midpoint knot value* hereafter, since point $\mathbf{P}_2$ is always in the middle of the two endpoints in terms of the sequence of data points). Therefore, point $\mathbf{P}_2$ is the value of function $\mathbf{m}$ evaluated at u such that the parameter u splits the parameter s by the division ratio of $u : (1-u)$. Hence,

$$\mathbf{m}(0) = \mathbf{P}_1, \quad \mathbf{m}(u) = \mathbf{P}_2, \quad \mathbf{m}(1) = \mathbf{P}_3$$

That is, we think of point $\mathbf{P}_2$ as the point mapped from u value of the parameter s domain, so that

$$a = 0 \text{ at } s = 0$$
$$a = 1 \text{ at } s = u$$
$$b = 0 \text{ at } s = u$$
$$b = 1 \text{ at } s = 1$$

Assuming a linear relationship among three parameters, the relation can be represented as

$$a = k_1 s + k_2$$
$$b = k_3 s + k_4$$

The constants k_i of these equations can be determined by applying the above four constraints. It follows that

$$a = \frac{1}{u} s$$
$$b = \frac{1}{1-u} s - \frac{u}{1-u}$$

Therefore, the quadratic curve incorporating the new parameter u of the midpoint knot value is,

$$\begin{aligned} \mathbf{m}(s) &= (1-s)\, \mathbf{f}\,(a) + s\, \mathbf{g}\,(b) \\ &= (1-s)\,((1-a)\mathbf{P}_1 + a\mathbf{P}_2) + s\,((1-b)\mathbf{P}_2 + b\mathbf{P}_3) \\ &= [s^2 \; s \; 1] \begin{bmatrix} \frac{1}{u} & -\frac{1}{u}-\frac{1}{1-u} & \frac{1}{1-u} \\ -1-\frac{1}{u} & \frac{1}{u}+\frac{1}{1-u} & 1-\frac{1}{1-u} \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} \end{aligned} \quad (3\,1)$$

The quadratic curve of the ordinary cardinal splines belongs to the special case of this class of quadratics where the midpoint knot parameter u becomes $\frac{1}{2}$ Notice that the above quadratic has two different parameters The parameter s is the original parameter which varies between zero and one to make up a curve, while the parameter u represents the location of the midpoint knot values leading to the variation of the entire curve shape.

We are now in a position to blend two of these quadratics to form a cubic spline

As in Figure 3-5, consider two parametric quadratics **m**(a) and **n**(b), each defined in terms of parameters a and b respectively, where a and b are in the range of $(0 \leq a, b \leq 1)$ Assume that **m**(a) is a quadratic which interpolates three points $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$, such that the parameter a varies between zero and one while the quadratic **m**(a) makes up the curve from $\mathbf{P}_1$ to $\mathbf{P}_3$ Similarly, the parameter b runs from zero to one inclusively while the quadratic **n**(b) shapes the curve from $\mathbf{P}_2$ to $\mathbf{P}_4$ Also assume that the quadratic curves **m**(a) and **n**(b) are formulated with the midpoint knot parameters u and v respectively, so that point $\mathbf{P}_2$ is the value of function **m** evaluated at u, and point $\mathbf{P}_3$ is the value of function **n** evaluated at v Therefore, the parameter u splits the parameter a by a division ratio of u : (1-u), while the parameter v splits the parameter b by a division ratio of v : (1-v) Hence,

$$\mathbf{m}(0) = \mathbf{P}_1 \qquad \mathbf{m}(u) = \mathbf{P}_2 \qquad \mathbf{m}(1) = \mathbf{P}_3$$

$$\mathbf{n}(0) = \mathbf{P}_2 \qquad \mathbf{n}(v) = \mathbf{P}_3 \qquad \mathbf{n}(1) = \mathbf{P}_4$$

Our aim is to derive an expression for the interpolating curve **p**(s) which is defined in the interval $[\mathbf{P}_2, \mathbf{P}_3]$ We rewrite the individual quadratic parabola,

$$\mathbf{m}(a) = [a^2 \ a \ 1] \, \mathbf{B} \, [\mathbf{P}_1 \ \mathbf{P}_2 \ \mathbf{P}_3]^T$$

with matrix

$$\mathbf{B} = \begin{bmatrix} \frac{1}{u} & -\frac{1}{u}-\frac{1}{1-u} & \frac{1}{1-u} \\ -1-\frac{1}{u} & \frac{1}{u}+\frac{1}{1-u} & 1-\frac{1}{1-u} \\ 1 & 0 & 0 \end{bmatrix}$$

and

$$\mathbf{n}(b) = [b^2 \ b \ 1] \ \mathbf{C} \ [\mathbf{P}_1 \ \ \mathbf{P}_2 \ \ \mathbf{P}_3]^T$$

with matrix

$$\mathbf{C} = \begin{bmatrix} \frac{1}{v} & -\frac{1}{v}-\frac{1}{1-v} & \frac{1}{1-v} \\ -1-\frac{1}{v} & \frac{1}{v}+\frac{1}{1-v} & 1-\frac{1}{1-v} \\ 1 & 0 & 0 \end{bmatrix}$$

If $\mathbf{p}(s)$ is a linear blending of $\mathbf{m}(a)$ and $\mathbf{n}(b)$, it can be represented as

$$\mathbf{p}(s) = (1-s)\,\mathbf{m}(a)+s\,\mathbf{n}(b)$$

The linear relationship among parameters a b and s is expressed as

$$a = k_1 s+k_2$$

$$b = k_3 s+k_4$$

where the constants k_i are yet to be determined The constraints between the parameters are:

$a = u$ at $s = 0$

$a = 1$ at $s = 1$

$b = 0$ at $s = 0$

$b = v$ at $s = 1$

Applying these constraints to the above equations, we get

$a = (1-u)\,s+u$, and

$b = v\,s.$

Consequently,

$$p(s) = (1-s)\,\mathbf{m}(a)+s\,\mathbf{n}(b)$$
$$= (1-s)\,[a^2 \;\; a \;\; 1]\,\mathbf{B}+s\,[b^2 \;\; b \;\; 1]\,\mathbf{C}$$
$$= (1-s)\,[((1-u)s+u)^2 \quad (1-u)s+u \quad 1]\,\mathbf{B}+s\,[(vs)^2 \quad vs \quad 1]\,\mathbf{C}$$

Substitution of the matrices **B** and **C** into this equation yields cubic spline p(s) in terms of the midpoint knot parameters, u and v. Hence we can define a new class of cardinal splines as follows

Definition The free form of cardinal splines (abbreviated as the *free form splines* hereafter) are defined analytically as

$$\mathbf{p}(s) = [s^3 \;\; s^2 \;\; s \;\; 1]\,\mathbf{A}\,\{\mathbf{P}_1 \;\; \mathbf{P}_2 \;\; \mathbf{P}_3 \;\; \mathbf{P}_4\}^T$$

with

$$\mathbf{A} = \begin{bmatrix} -u+2-\frac{1}{u} & -1+\frac{1}{u}+v & u+\frac{1}{v-1} & -v-1+\frac{1}{1-v} \\ 2u-4+\frac{2}{u} & 2-\frac{2}{u}-v & -2u+1+\frac{1}{1-v} & v+1-\frac{1}{1-v} \\ -u+2-\frac{1}{u} & -2+\frac{1}{u} & u & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad (3.2)$$

where

$\mathbf{P}_1$ $\mathbf{P}_2$ $\mathbf{P}_3$ and $\mathbf{P}_4$ are four consecutive data points to be interpolated,

u, v are the midpoint knot parameters corresponding to point $\mathbf{P}_2$ and $\mathbf{P}_3$ respectively

s is the parameter varying between zero and one, while the curve p(s) ranges from point $\mathbf{P}_2$ to point $\mathbf{P}_3$.

The free form splines include the ordinary cardinal splines as a subset in which the parameter values are, $u = v = \frac{1}{2}$

Notice that the curve shape in the interval $[\mathbf{P}_2,\mathbf{P}_3]$ is determined by a set of u, v parameters while that of the interval $[\mathbf{P}_3,\mathbf{P}_4]$ is determined by another set of u, v parameters. The parameter v may indicate the knot value of the midpoint $\mathbf{P}_3$ during the evaluation of the curve within $[\mathbf{P}_2,\mathbf{P}_3]$ Suppose we are evaluating the curves in the next interval $[\mathbf{P}_3,\mathbf{P}_4]$ by introducing another point $\mathbf{P}_5$ following $\mathbf{P}_4$. This time, the knot value of the midpoint $\mathbf{P}_3$ should be represented as u, and the knot value of the midpoint $\mathbf{P}_4$ takes the role of v Therefore, for each successive evaluation of piecewise splines, midpoint knot parameters associated with each data point take the role of u and v, in turn

Instead of saying that the free form splines are a linear blending of quadratics, one can regard the free form splines as a cubic blending of data points. The cubic blending function, in this sense, is the multiplication of the matrix $[s^3\ s^2\ s\ 1]$ and **A** Consequently, the elements of the matrix $[s^3\ s^2\ s\ 1]$ multiplied by the matrix **A** compose the four basis functions [Bars86, Gor74b] of the free form splines Notice that the sum of each row of matrix **A** is zero except for the fourth row, which is one This meets the normalization constraints [Cat74] required for the basis function.

It is most important to note that our derivation is strictly analytic so that the splines can be represented by a single four by four matrix It is this simplicity of representation that enables easy implementation of the spline functions in various design applications

Summarizing, we started out with the linear blending of two points During the blending of the two consecutive linear segments, an extra degree of freedom is found in assigning the knot values corresponding to the midpoints. Based on the observation that the location of the knot values affects the shape of the quadratic parabola, we incorporated the midpoint knot values as new parameters into the formulation of the quadratic parabola. These parabolas, in turn, are blended linearly to yield the cubic splines defined as the free form splines.

Mathematical Description of the G^1 Free Form Splines

In the previous section, we created a new class of interpolating splines called free form splines. In an interactive design environment, the curve shapes which can be produced by free form splines are almost unlimited. This variety mainly stems from the fact thet the left tangents and the right tangents are defined and controlled separately so thet the slopes can be discontinuous at data points. However, we do not want to destroy slope continuity at data points as our objective. Since what we aim for is curves which are wiggle-free within intervals between given points, the splines interpolating backbones or the motion path must remain visually smooth at the data points.

In this section, we extend the free form splines to an important class of splines called G^1 splines, to assure visual smoothness at data points. This is possible by differentiating the expression for free form splines and by imposing some constraints on the tangents thus generated. Based on the observation of the limitation of the tangents of the ordinary cardinal splines, as a special case of free form splines, the concept of visual continuity is incorporated into the free form splines. A restrictive condition is claimed and verified for the free form splines to be visually continuous.

The free form splines belong to one of the widest class of cardinal splines. The term "wide," in this sense, means the variety of the curve shapes which can be generated by the splines. By varying the midpoint knot parameters (i.e., u and v), we can acquire numerous curve shapes interpolating given data points. From the animator's point of view, such variety leads to diversity in planning the motion path between data points. On the other hand, artists or designers may use the variety to generate a multitude of curve shapes passing through the given data points in a static scene. However, this variety is useful only if we can control it.

The principal difference between free form splines and ordinary cardinal splines can be described in terms of the parametric derivatives at the data points. In each data point, the left and right tangents of ordinary cardinal splines always agree in both direction and magnitude. In contrast, free form splines distinguish the left tangent from the right tangent, and the direction and magnitude of the left tangent may differ from those of the right tangent.

The tangential properties of the free form splines can be described by fundamental calculus. Suppose we are interested in the evaluation of the tangents at point $\mathbf{P}_3$ (see Figure 3-5 for example.) To estimate the curves in the interval $[\mathbf{P}_2,\mathbf{P}_3]$, we need to know midpoint knot parameters u_1 and v_1 corresponding to the points $\mathbf{P}_2$ and $\mathbf{P}_3$, respectively. Taking the parametric derivative of the free form splines, we obtain

$$\frac{dp(s)}{ds} = [3s^2\ 2s\ 1\ 0]\,\mathbf{A}\ [\mathbf{P}_1\ \mathbf{P}_2\ \mathbf{P}_3\ \mathbf{P}_4]^T$$

with

$$\mathbf{A} = \begin{bmatrix} -u_1+2-\frac{1}{u_1} & -1+\frac{1}{u_1}+v_1 & u_1+\frac{1}{v_1-1} & -v_1-1+\frac{1}{1-v_1} \\ 2u_1-4+\frac{2}{u_1} & 2-\frac{2}{u_1}-v_1 & -2u_1+1+\frac{1}{1-v_1} & v_1+1-\frac{1}{1-v_1} \\ -u_1+2-\frac{1}{u_1} & -2+\frac{1}{u_1} & u_1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \quad (3.3)$$

Hence the left tangent at point P_3 is given by

$$\left.\frac{dp(s)}{ds}\right|_{s=1} = \left[0 \quad v_1-1 \quad 2-\frac{1}{1-v_1} \quad -(v_1+1)+\frac{1}{1-v_1}\right] [\mathbf{P}_1 \ \mathbf{P}_2 \ \mathbf{P}_3 \ \mathbf{P}_4]^T$$

By introducing another point $\mathbf{P}_5$ next to point $\mathbf{P}_4$, we can derive the right tangent at point $\mathbf{P}_3$. If the two midpoint knot parameters corresponding to $\mathbf{P}_3$ and $\mathbf{P}_4$ are denoted as u_2 and v_2 respectively, the right tangent of point $\mathbf{P}_3$ is expressed as,

$$\left.\frac{dq(s)}{ds}\right|_{s=0} = \left[-u_2+2-\frac{1}{u_2} \quad -2+\frac{1}{u_2} \quad u_2 \quad 0\right] [\mathbf{P}_2 \ \mathbf{P}_3 \ \mathbf{P}_4 \ \mathbf{P}_5]^T$$

where

$\mathbf{q}(s)$ represents the spline in the interval $[\mathbf{P}_3,\mathbf{P}_4]$

Let us describe the ordinary cardinal splines as a special case of this free form spline. The ordinary cardinal splines replace both midpoint knot parameters v_1 and u_2 with $\frac{1}{2}$ so that the left and right tangents agree. That is,

$$\left.\frac{dp(s)}{ds}\right|_{s=1} = \left.\frac{dq(s)}{ds}\right|_{s=0} = \frac{1}{2}\,(\mathbf{P}_4-\mathbf{P}_2)$$

Geometrically speaking, the left and right tangents of the ordinary cardinal spline have the same magnitude and direction at point $\mathbf{P}_3$. It is this correspondence of the left and right tangents in the ordinary cardinal spline that is said to be C^1 continuous. More specifically, the tangent at point P_3 is one half the vector (P_4-P_2) Therefore, the magnitude and direction of the tangential vectors are fixed, once the data points are given; there is no way to control the magnitude and direction of the tangential vectors As a result, the ordinary cardinal splines are often estimated to be unduly strained [Man74]

In contrast, the tangents of the free form spline can vary depending on the selected midpoint knot parameters, as can be seen by the above equations for the tangents As a matter of fact, one can assign two different knot parameters (v_1 and u_2) for a single midpoint P_3, and produce curves with slope discontinuity at that point. In other words, the free form splines include those curves which have slope discontinuity (*C^0 continuity*)

Although the free form splines are developed such that they could incorporate flexible tangents at data points, the same flexibility might hamper the smoothness Splines having two different tangential directions at a data point cannot be said to be smooth. How do we impose the smoothness criterion on the free form splines while avoiding the unduly strained, tangential constraints of the ordinary cardinal splines?

Geometric continuity [Bart87, Bars88] addresses the problem of parametric continuity. Parametric continuity means that the left and right derivatives with respect to the knot parameter are identical. However, as was mentioned in Chapter 2, the parametric continuity does not necessarily agree with the visual continuity The reason is fairly straightforward Saying that the parametric derivatives are identical means that the derivatives of individual x, y, z components of the Cartesian coordinates *with respect to the parameter* is the same for left and right tangents Nevertheless, the visual perception is based on the continuity of the tangential directions which are represented by a relationship between such tangents Therefore, the visual continuity, often referred to as geometric continuity, is preserved if the direction of the unit tangent is identical at a data point.

In light of the degree of the versatility in the curve shapes, C^0 of the free form splines covers the widest class of the splines including discontinuous curves The G^1 splines lie in the middle of C^0 and C^1 in terms of the degree of the versatility They are not as restricted as the C^1 splines, since they relax the tangent constraints of the ordinary cardinal splines. They allow the left and right tangents to vary *in magnitude* as long as both tangents have the same unit tangent vectors Moreover, the direction of the tangents of the G^1 splines are not limited to the the vector from the previous point to the next point

By definition [Bart87, Bars88], the geometric first-order derivative is defined to be continuous (G^1 continuous) if

$$\text{Unit tangent of } \left[\left. \frac{dp(s)}{ds} \right|_{s=1} \right] = (k) \text{ Unit tangent of } \left[\left. \frac{dq(s)}{ds} \right|_{s=0} \right]$$

where

k is an arbitrary multiplication constant

Figure 3-6 shows two different midpoint knot parameters v_1 and u_2 assigned to point P_3 During the evaluation of the curve in the interval $[P_2,P_3]$, v_1 takes the role of midpoint knot parameter so that the parabola A will be blended Similarly, the parameter u_2 will make the parabola B be blended during the evaluation of the curve in the interval $[P_3,P_4]$. In practice, every data point can be associated with two different midpoint knot values The net effect of this dual definition of the midpoint knot values is the ability to change the direction of the tangent at the data points For example in the final cubic free form splines, Dv_1 is the incoming tangent and Du_2 is the outgoing tangent at point $\mathbf{P}_3$.

Noticing that the G^1 splines establish visual continuity while avoiding the tangent constraints of ordinary cardinal splines, we now impose certain constraints on the free form splines to make them G^1 continuous.

Claim:

Define four arbitrary midpoint knot parameters u_1, v_1, u_2, v_2 corresponding to points $\mathbf{P}_2$, $\mathbf{P}_3$, $\mathbf{P}_3$ and $\mathbf{P}_4$ as in Figure 3-6, such that point $\mathbf{P}_3$ has two midpoint knot values associated with it. Then the free form splines are G^1 continuous if

$$v_1 = u_2$$

That the midpoint knot parameters are consistent for the evaluation of both the left and right splines is a sufficient condition for the free form splines to be G^1 continuous. In terms of Figure 3-6, the free form splines will be G^1 continuous if either the quadratic parabola A or B is used consistently, to evaluate the cubic splines in both the intervals $[\mathbf{P}_2, \mathbf{P}_3]$ and $[\mathbf{P}_3, \mathbf{P}_4]$.

Proof:

Without loss of generality, we may treat an addition of the data points as directional vectors. The left tangent of point $\mathbf{P}_3$ is,

$$\left.\frac{dp(s)}{ds}\right|_{s=1} = (v_1-1)\,\mathbf{P}_2+(2+\frac{1}{v_1-1})\,\mathbf{P}_3+(-(v_1+1)+\frac{1}{1-v_1})\,\mathbf{P}_4$$

$$= (v_1-1)\,\mathbf{P}_2+(1+\frac{1}{v_1-1}+1)\,\mathbf{P}_3+(-(v_1+1)+\frac{1}{1-v_1})\,\mathbf{P}_4$$

$$= (\frac{1}{1-v_1}-1)\,(\mathbf{P}_4-\mathbf{P}_3)-v_1\,(\mathbf{P}_4-\mathbf{P}_3+\mathbf{P}_3-\mathbf{P}_2)+(\mathbf{P}_3-\mathbf{P}_2)$$

$$= \frac{v_1^2}{1-v_1}(\mathbf{P}_4-\mathbf{P}_3)+(1-v_1)(\mathbf{P}_3-\mathbf{P}_2). \tag{3.4}$$

The right tangent at the same point is,

$$\left.\frac{dq(s)}{ds}\right|_{s=0} = (-u_2+2-\frac{1}{u_2})\,\mathbf{P}_2+(-2+\frac{1}{u_2})\,\mathbf{P}_3+u_2\,\mathbf{P}_4$$

$$= u_2\,(\mathbf{P}_4-\mathbf{P}_2)+(-2+\frac{1}{u_2})\,(\mathbf{P}_3-\mathbf{P}_2)$$

$$= u_2\,(\mathbf{P}_4-\mathbf{P}_3+\mathbf{P}_3-\mathbf{P}_2)+(-2+\frac{1}{u_2})\,(\mathbf{P}_3-\mathbf{P}_2)$$

$$= u_2\,(\mathbf{P}_4-\mathbf{P}_3)+\frac{(1-u_2)^2}{u_2}(\mathbf{P}_3-\mathbf{P}_2) \tag{3.5}$$

If

$$v_1 = u_2 = k$$

then

$$\left.\frac{dq(s)}{ds}\right|_{s=0} = \left.\frac{dp(s)}{ds}\right|_{s=1}\left(\frac{1-k}{k}\right)$$

Since the right tangent is a scalar multiple of the left tangent by the factor of $\left(\frac{1-k}{k}\right)$ both tangents have the same unit tangent vector. This completes the proof

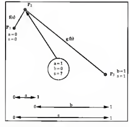

Figure 3-3 Linear blending of two line segments The line segments **f** (a) and **g** (b) are also linear blending of two data points $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_2$, $\mathbf{P}_3$ respectively A quadratic in the interval $[\mathbf{P}_1, \mathbf{P}_3]$ can be produced by varying the normalized parameter s from zero to one inclusively

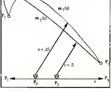

Figure 3-4 Variation of the knot values corresponding to the midpoint P_2. The curve $\mathbf{m}_1$ and $\mathbf{m}_2$ are generated by inserting the midpoint knot parameter value of .25 and .5, respectively

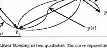

Figure 3-5. Linear blending of two quadratics. The curve represented by P(s) is the cubic spline resulting from the blending of the quadratics m(a) and n(b). It is defined in the region $[P_2,P_3]$

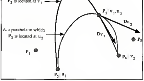

Figure 3-6 The tangents caused by two different midpoint knot parameters defined at a single point $\mathbf{P}_3$ $\mathbf{Dv}_1$ represents the tangent at point $\mathbf{P}_3$ when its midpoint knot parameter is consistently equal to $\mathbf{v}_1$, throughout the interpolation of the region $[\mathbf{P}_1, \mathbf{P}_2]$ and $[\mathbf{P}_2, \mathbf{P}_3]$ Similarly for $\mathbf{Du}_2$ If the midpoint knot parameter has dual value at thet point, the left and right tangents do not match as shown

Vector Analysis of the G^1 Free Form Splines

In the previous section, we derived the relation between the left and right tangents of the G^1 free form splines. The free form splines, originally devised to enable wiggle-free interpolation has to maintain consistent midpoint knot parameter values, to assure visual smoothness at data points. Now we proceed to designate the exact values of the knot parameters to finally enable wiggle-free interpolation. If it is an interactive design environment, the parameters, and hence the curve shapes, can be varied until a satisfactory one is found. However, one prefers to automatically drawing the backbone lines or the motion path empirically to interruptions by the designer's interaction. We do not want the flexibility to redraw the backbone lines iteratively until the curve shapa is satisfactory for our animation system.

This section analyzes the G^1 free form splines by an extensive usage of vector calculus. Based upon the characteristic behavior of individual curves, three types of G^1 splines will be developed. However, tha application of G^1 splines is not limited to these categories. Almost limitless kinds of G^1 curve shapes can be produced in our approach, by varying a characteristic constant d, which will be explained shortly.

G^0 Continuity

Before dalving into the G^1 free form splines, let us digress briefly into the G^0 free form splines to emphasize that a curve can be controlled by noting the component vectors.

In general, the free form splines can be said to be G^0 continuous in terms of continuity condition at the data points. The left and right tangents are not of the same direction at a data point, as can be evidenced by Equation 3.4 and 3.5. Therefore, in the general free form splines, the restrictive condition that regulates the ordinary cardinal splines has been removed.

One of the advantages of separate control of the left and right tangents is that of being able to make an intentional cusp. For instance, the direction of a bouncing ball [Koc84] shows discontinuity near the contact point, and sometimes an animator needs to simulate a motion path for such movements. More specifically, the motion path at the contact point is continuous but the tangents at that point are discontinuous. This sudden directional change can easily be incorporated into the free form splines with proper control of the midpoint knot parameters. With deliberate assignment of these parameters, we could control even the level of the discontinuity.

Figure 3-7 illustrates the discontinuity effect. The curves marked C1, C2, and C3 represent three different instances of continuity variation, caused by assigning dual midpoint knot parameter values to point P_3. The symmetrical tangents at point P_3 can be explained by the mathematical derivation used in the last section. From Equation 3.4 and 3.5, we have

$$\text{left tangent} = \left.\frac{dp(s)}{ds}\right|_{s=1} = \frac{v_1^2}{1-v_1}(P_4-P_3)+(1-v_1)(P_3-P_2)$$

$$\text{right tangent} = \left.\frac{dq(s)}{ds}\right|_{s=0} = u_2(P_4-P_3)+\frac{(1-u_2)^2}{u_2}(P_3-P_2)$$

If we let

$$v_1+u_2 = 1,$$

the (P_4-P_3) component of the left tangent is the same as the (P_3-P_2) component of the right tangent. Similarly, the (P_3-P_2) component of the left tangent is identical to the (P_4-P_3) component of the right tangent. Since the magnitudes of the two vectors are identical, and since the magnitudes of components vectors are symmetrical each other, the resulting tangent vectors are symmetrical with respect to point P_3.

In practice, there can be numerous control strategies that could produce various shapes, and this example shows that the analysis of the component vectors can be a useful tool for the design and control of motion path employing the free form splines.

Type I G¹ Free Form Splines

Let us rewrite the expression for the tangents of the G^1 free form splines with different notation, such that

$$\mathbf{D}_L = \frac{k^2}{1-k}(P_4-P_3)+(1-k)(P_3-P_2)$$

$$\mathbf{D}_R = k(P_4-P_3)+\frac{(1-k)^2}{k}(P_3-P_2) \tag{3.6}$$

where

$\mathbf{D}_L$ is the left tangent at point P_3,

$\mathbf{D}_R$ is the right tangent at point P_3,

k is the knot parameter corresponding to midpoint P_3.

Notice that the G^1 free form splines have only a single knot parameter associated with each data point, while the free form splines have two knot parameters for each data point.

As was seen in the last section, the magnitude of the right tangent is a scalar multiple of the left tangent by the factor of $\left(\frac{1-k}{k}\right)$. For example, if the midpoint knot parameter k decreases, the multiplication factor increases, and the magnitude of the right tangent becomes greater. Physically, the consequence of the increase in tangential magnitude means that the curve tends to persist in a certain direction for longer intervals On the other hand, if the midpoint knot parameter *k* increases, the curve to the right of the data point quickly loses the inclination to persist in the previous direction and tends to form an inflection toward the next data point as soon as it leaves the data point. Therefore, it is possible to control the inflection by a proper selection of the midpoint knot parameter *k*.

However, the curve shapes are determined not only by the magnitude of the tangents, but also by the direction of the tangents The information about the magnitude of the tangents alone is not sufficient to shape the curves

Both the slope vectors $\mathbf{D}_L$ and $\mathbf{D}_R$ can be decomposed into two components: the component in the direction of the vector $(\mathbf{P}_4\text{-}\mathbf{P}_3)$ and the component in the direction of the vector $(\mathbf{P}_3\text{-}\mathbf{P}_2)$ Figure 3-8 shows the decomposition of the right slope vector into individual components The vector $(\mathbf{P}_3\text{-}\mathbf{P}_2)$ is translated to point $\mathbf{P}_3$ in order to facilitate the addition of the two component vectors. The vector $\mathbf{D}_{Ra}$ is the component of the right tangent in the direction of the vector $(\mathbf{P}_3\text{-}\mathbf{P}_2)$, and similarly for the vector $\mathbf{D}_{Rb}$. These vectors are combined to form the right tangent vector $\mathbf{D}_R$

Therefore, the relationship between the coefficients of the component vectors $(\mathbf{P}_4\text{-}\mathbf{P}_3)$ and $(\mathbf{P}_3\text{-}\mathbf{P}_2)$, as determined from the midpoint knot parameter value *k*, plays a critical role in controlling the tangential directions of G^1 free form splines.

Let the midpoint knot parameters be a function of the magnitudes of the vectors $(\mathbf{P}_3\text{-}\mathbf{P}_2)$ and $(\mathbf{P}_4\text{-}\mathbf{P}_3)$. That is, the values of the midpoint knot parameters are constrained such that they are dependent on the distances between the neighboring points. Then Type I spline is defined such that the dependency can be expressed as,

$$\begin{aligned} \mathbf{P}_3\text{-}\mathbf{P}_2 &= \frac{k}{1\text{-}k}\ t\ \mathbf{a} \\ \mathbf{P}_4\text{-}\mathbf{P}_3 &= \frac{1\text{-}k}{k}\ t\ \mathbf{b} \end{aligned} \tag{3.7}$$

where

vectors **a**, **b** denote the unit vectors to the direction of the vectors $(\mathbf{P}_3\text{-}\mathbf{P}_2)$ and $(\mathbf{P}_4\text{-}\mathbf{P}_3)$ (referred to as *component vectors*, hereafter) respectively

Dividing the equations, we get

$$\left|\frac{\mathbf{P}_3\text{-}\mathbf{P}_2}{\mathbf{P}_4\text{-}\mathbf{P}_3}\right| = \left(\frac{k}{1\text{-}k}\right)^2$$

If we define the knot division ratio as *k* (1-*k*), then the square of the knot division ratio is equal to the chord length ratio between data points Therefore, the midpoint knot parameter *k*, and the scaling constant t, in terms of the chord length are

$$k = \frac{1}{1+\sqrt{\left|\frac{\mathbf{P}_4\text{-}\mathbf{P}_3}{\mathbf{P}_3\text{-}\mathbf{P}_2}\right|}}$$

$$t = \sqrt{\left|(\mathbf{P}_3\text{-}\mathbf{P}_2)\,(\mathbf{P}_4\text{-}\mathbf{P}_3)\right|} \tag{3.8}$$

If Equation 3.7 is inserted into Equation 3.6,

$$\begin{aligned} \mathbf{D}_L &= (1\text{-}k)\,(\mathbf{P}_3\text{-}\mathbf{P}_2)+\frac{k^2}{1\text{-}k}\,(\mathbf{P}_4\text{-}\mathbf{P}_3) \\ &= kt\,\mathbf{a}+kt\,\mathbf{b} \end{aligned}$$

$$\mathbf{D}_R = \frac{(1\text{-}k)^2}{k}\,(\mathbf{P}_3\text{-}\mathbf{P}_2)+k\,(\mathbf{P}_4\text{-}\mathbf{P}_3)$$

$$= (1-k)t\,\mathbf{a}+(1-k)t\,\mathbf{b} \qquad (3.9)$$

Hence, the component vectors of the left tangent $\mathbf{D}_L$ are of identical magnitude. The same holds true for the right tangent $\mathbf{D}_R$, except that they are now scaled by a factor of $\left(\frac{1-k}{k}\right)$.

A geometrical consequence of the above expression for Type I spline is that the tangential direction is the bisection of the two unit vectors **a** and **b**, as shown in Figure 3-8. The property that the tangential directions are identical, is an expected result, since the Type I spline is G^1 continuous.

Moreover, the magnitude ratio of the left tangent to the right tangent is proportional to the knot division ratio *k* : (1-*k*).

The advantage of the Type 1 spline can be explained by comparing it with the curve shapes of the ordinary cardinal splines. Figure 3-9 shows the slope vectors of the ordinary cardinal splines. The vectors $\mathbf{D}_L$ and $\mathbf{D}_R$ denote the left and right tangents at point $\mathbf{P}_3$, respectively. Note that the magnitude and direction of the tangents determine the shape of the curves in the interval $[\mathbf{P}_2,\mathbf{P}_3]$ and $[\mathbf{P}_3,\mathbf{P}_4]$. As can be noticed, the direction of the vectors $\mathbf{D}_L$ and $\mathbf{D}_R$ is identically set to the vector $(\mathbf{P}_4$-$\mathbf{P}_2)$ in ordinary cardinal splines. In other words, the tangent vector is one half the sum of the two component vectors $(\mathbf{P}_3$-$\mathbf{P}_2)$ and $(\mathbf{P}_4$-$\mathbf{P}_3)$. Since the vector $(\mathbf{P}_4$-$\mathbf{P}_3)$ is *greater* in magnitude, the sum tends to resemble the vector $(\mathbf{P}_4$-$\mathbf{P}_3)$. Being influenced by the major component vector, the curve shows a bend *before* it reaches point $\mathbf{P}_3$. Such bending itself does not cause any problem as long as it maintains smooth transition near point $\mathbf{P}_3$.

What matters, and causes the wiggles, is the magnitude of the tangent vectors. Figure 3-10 shows two extreme cases of the ordinary cardinal splines. The curve marked as *c2* shows the behavior of the ordinary cardinal splines when point $\mathbf{P}_4$ moves away from $\mathbf{P}_3$ to the direction of the vector $(\mathbf{P}_4\text{-}\mathbf{P}_3)$ As the distance between the two points $\mathbf{P}_3$ and $\mathbf{P}_4$ increases, the magnitude of both tangents increases since

$$\mathbf{D}_L = \mathbf{D}_R = \frac{1}{2}\,(\mathbf{P}_4\text{-}\mathbf{P}_2)$$

in the ordinary cardinal splines Vector $\mathbf{D}_{c2}$ in Figure 3-10 represents these tangents With point $\mathbf{P}_4$ moving farther away, the curve tends to prepare for directional change well before it reaches point $\mathbf{P}_3$, since the magnitude of the *left* tangent at $\mathbf{P}_3$ should match one half the distance between $\mathbf{P}_4$ and $\mathbf{P}_2$ However, the interval assigned for that directional change (i.e., $[\mathbf{P}_2,\mathbf{P}_3]$) is so narrow in this case that curve cannot avoid a wiggle or a loop In contrast, the vector $\mathbf{D}_{c1}$ (the tangents when point $\mathbf{P}_4$ is drawn near $\mathbf{P}_3$) exhibits a reduction in magnitude, and thus reduction in magnitude of the left tangent makes the curve *c1* linear An extreme case of the reduction is that the left and the right tangents are identically zero, so that the splines become a pair of straight line segments joined at the point $\mathbf{P}_3$

Figure 3-11 illustrates the curve shape of Type I spline for the same data points as in Figure 3-9 The direction of tangents is the average of two unit vectors **a**, **b** so thet it could bisect the unit vectors. Note that the left and right tangents of the Type I spline have different magnitudes The same is true for Type II and Type III splines to be developed shortly Compare Figure 3-11 with Figure 3-9 in the interval $[\mathbf{P}_2,\mathbf{P}_3]$ The curve in Figure 3-11 is relatively flat compared with that of Figure 3-9 in this interval so that the tendency to wiggle is alleviated This is so because the magnitude of the left tangent $\mathbf{D}_L$ can be reduced while thet of the right tangent increases. This tendency of the Type I spline to adapt itself to the relative distance can be explained mathematically

The tangents of Type 1 splines in terms of the distance between data points can be derived from Equation 3.8 and 3.9 It follows that

$$|\mathbf{D_L}| = \sqrt{2}\,k\,t$$
$$|\mathbf{D_R}| = \sqrt{2}\,(1-k)\,t$$

with

$$k\,t = \frac{\sqrt{|(\mathbf{P}_3-\mathbf{P}_2)(\mathbf{P}_4-\mathbf{P}_3)|}}{1+\sqrt{\left|\frac{\mathbf{P}_4-\mathbf{P}_3}{\mathbf{P}_3-\mathbf{P}_2}\right|}}$$

$$(1-k)\,t = \frac{|\mathbf{P}_4-\mathbf{P}_3|}{1+\sqrt{\left|\frac{\mathbf{P}_4-\mathbf{P}_3}{\mathbf{P}_3-\mathbf{P}_2}\right|}} \qquad (3\ 10)$$

As the vector $(\mathbf{P}_3-\mathbf{P}_2)$ decreases, the value of kt will decrease so that the magnitude of the left tangent is reduced Meanwhile, the magnitude of the right tangent increases as can be verified by examining Equation 3 10 Therefore, the magnitude of the *left* tangent at point $\mathbf{P}_3$ becomes proportional to the distance between $\mathbf{P}_2$ and $\mathbf{P}_3$ in the Type I spline. It is this adaptability of the tangential magnitude that removes the wiggles, which typically appear when a curve in a short interval has to adapt to tangents with excessive magnitude

Comparing Figures 3-9 and 3-11 in the interval $[\mathbf{P}_3,\mathbf{P}_4]$, we notice that the Type 1 spline has slightly more swing in that region This swing is a visual clue that the magnitude of the right tangent of the Type 1 spline is greater than that of the ordinary cardinal spline due to the tendency of the Type I spline to be inflective in the longer interval

In summary, the division ratio of the midpoint knot parameters of Type I splines is made to be proportional to the square root of the chord lengths among data points Compared with the ordinary cardinal splines, Type 1 splines exhibit improvements in removing the wiggles in the shorter intervals, while showing more swings in the longer intervals Moreover, the tangential direction is set to be always bisecting the two unit vectors formed by the three consecutive data points.

Type II G^1 Free Form Splines

Type II splines encompass those splines whose midpoint knot parameters have the following relationship with the chord length between data points

$$P_3\text{-}P_2 = (k)\ t\ a$$
$$P_4\text{-}P_3 = (1\text{-}k)\ t\ b \qquad (3\ 11)$$

where

the vectors a b denote the unit vectors to the direction of the vectors $(P_3\text{-}P_2)$ and $(P_4\text{-}P_3)$, respectively

In other words, the midpoint knot parameter *k* is constrained by

$$\left|\frac{P_3\ P_2}{P_4\ P_3}\right| = \left(\frac{k}{1\ k}\right)$$

so that the midpoint knot division ratio is the same as the rate of the chord length This distinguishes Type II splines from the Type 1 splines Solving Equation 3 11, we get the midpoint knot value *k* and the scaling constant t as,

$$k = \frac{1}{1+\left|\frac{P_4\ P_3}{P_3\text{-}P_2}\right|}$$
$$t = |P_3\text{-}P_2| + |P_4\ P_3| \qquad (3\ 12)$$

Substitution of Equation 3 11 into Equation 3 6 yields the expression for the left and the right tangents as,

$$\mathbf{D_L} = (1-k)(\mathbf{P}_3-\mathbf{P}_2)+\frac{k^2}{1-k}(\mathbf{P}_4-\mathbf{P}_3)$$
$$= k(1-k)\,t\,\mathbf{a}+k^2\,t\,\mathbf{b}$$
$$\mathbf{D_R} = \frac{(1-k)^2}{k}(\mathbf{P}_3-\mathbf{P}_2)+k(\mathbf{P}_4-\mathbf{P}_3)$$
$$= (1-k)^2\,t\,\mathbf{a}+k(1-k)\,t\,\mathbf{b} \tag{3 13}$$

Notice that the right tangent is again the scalar multiple of the left tangent by the factor of $\left(\frac{1-k}{k}\right)$, guaranteeing the G^1 continuity

Turning our attention to the relationship between the component vectors making up the left and right tangents, we find that the magnitude ratio of the component vectors depends on the midpoint knot parameter *k*. An example of a Type II spline interpolating the same data points as in Figure 3-11 is illustrated in Figure 3-12. The rate of the chord length of the line segment $\mathbf{P}_2\mathbf{P}_3$ to that of the line segment $\mathbf{P}_3\mathbf{P}_4$ is 1 : 3.55 for this specific example, so that k = .22 by Equation 3.12.

The consequence of the *k* value is twofold:

1. It determines the magnitude ratio of the left tangent $\mathbf{D_L}$ to the right tangent $\mathbf{D_R}$ such that

$$\frac{\mathbf{D_L}}{\mathbf{D_R}} = \frac{k}{1-k} = .28$$

2. It determines the magnitude ratio of the directional components As can be seen in Figure 3-12, the rate of the **a** vector component to thet of the **b** vector component is,

$$\frac{\mathbf{a}\text{ component}}{\mathbf{b}\text{ component}} = \left|\frac{\mathbf{P}_4-\mathbf{P}_3}{\mathbf{P}_3-\mathbf{P}_2}\right| = \frac{1-k}{k} - 3.55$$

so thet the tangential direction is close to the **a** vector, a unit vector in the direction of (P_3-P_2)

The shorter the interval $[P_2,P_3]$, the larger the a component, and the higher weighting is given to the vector of the shorter intervals This holds true both for the left and right tangents, since the right tangent is a scalar multiple of the left tangent.

In essence, the rate of the chord length determines the midpoint knot parameter *k*, and the value of *k* determines the direction and the magnitude of the tangents

With respect to the absolute magnitude of the tangents, an interesting feature is found in Type II splines Inserting Equation 3 12 into Equation 3.13, we obtain

$$\begin{aligned} |D_L| &= k\,t\sqrt{k^2+(1-k)^2} \\ &= |P_3P_2|\sqrt{k^2+(1-k)^2} \end{aligned}$$

$$\begin{aligned} |D_R| &= (1-k)\,t\sqrt{k^2+(1-k)^2} \\ &= |P_4-P_3|\sqrt{k^2+(1-k)^2} \end{aligned} \qquad (3\ 14)$$

If *k* and (1-*k*) constitute the two sides of a right-angled triangle, then the magnitude of the left tangent is the length of the line segment P_2P_3 scaled by the hypotenuse of thet triangle

Compared with Figure 3-11, Figure 3-12 exhibits more flatness in the interval $[P_2,P_3]$, while it shows more reflection in the interval $[P_3,P_4]$ in the vicinity to the left of point P_3, the Type I curve is already changing its direction in anticipation of the requirement that the tangent should bisect the unit tangents of the component vectors at point P_3. In contrast, the Type II curve did not change its direction yet. Because the incoming vector (P_3-P_2) is still a dominant component of the tangent at point $\mathbf{P}_3$, the major change of the direction happens only after the curve passes through point $\mathbf{P}_3$

Notice the gradual change of tangential direction at point $\mathbf{P}_3$ in Figures 3-9, 3-11 and 3-12. The level that the tangent at $\mathbf{P}_3$ reflects the direction of the incoming vector $(\mathbf{P}_3\text{-}\mathbf{P}_2)$ is highest in Figure 3-12. The wiggles or loops can be regarded as the curve shape whose tangential direction undergoes a rapid change in a short interval. In view of the fact that such a sudden change of the tangential direction in the shorter interval is avoidable in Type II splines, it can be said that Type II splines are better than Type I splines for removing wiggles

In addition to these directional considerations, it is important to note that the magnitude of the left tangent versus right tangent plays another key role in eliminating the wiggles Written out explicitly,

$$\left|\frac{\mathbf{D}_L}{\mathbf{D}_R}\right|_{\text{ordinary cardinal splines}} = 1$$

$$\left|\frac{\mathbf{D}_L}{\mathbf{D}_R}\right|_{\text{Type I}} = \left|\frac{\mathbf{D}_L}{\mathbf{D}_R}\right|_{\text{Type II}} = \frac{k}{1\text{-}k} \tag{3 15}$$

For instance, the ratio of the tangential magnitude corresponding to Type II spline in Figure 3-12 is

$$\left|\frac{\mathbf{D}_L}{\mathbf{D}_R}\right|_{\text{Type II Fig. 3-12}} = \left.\frac{k}{1\ k}\right|_{k=.22} = .28$$

while that of Type 1 spline in Figure 3-11 is

$$\left|\frac{\mathbf{D}_L}{\mathbf{D}_R}\right|_{\text{Type I. Fig. 3-11}} = \left.\frac{k}{1\text{-}k}\right|_{k=.35} = .53$$

The smaller the ratio, the larger the reflection in the longer interval and the smaller the reflection (wiggle) in the shorter interval This can be verified by comparing Figure 3-11 with Figure 3-12 in the shorter interval $[P_2,P_3]$ and the longer interval $[P_3,P_4]$

In summary, Type II splines determine the midpoint knot parameter k such that the knot division ratio is the same as the rate of the chord length The tangent at the midpoint is dependent on the magnitudes of the vectors (P_3-P_2) and (P_4-P_3) In Type II splines, the parameter *k* acts to the tangential direction of the midpoints such that the direction of the shorter interval is weighted with larger values The same parameter *k* acts on the tangential magnitude such that the shorter interval has less magnitude. As a result of adjustments in both the direction and magnitude, the removal of wiggles in Type II splines is more pronounced than in Type I splines

Type III G^1 Free Form Splines

Because we already have defined and analyzed two types of visually continuous splines, we will describe the third type of splines as to be imbedded in a more general expression This general expression leads to the definition of a characteristic constant *d*

Type III splines are represented by the relation

$$P_3-P_2 = (1-k)^3 \; t \; \mathbf{a}$$
$$P_4-P_3 = (k)^3 \; t \; \mathbf{b} \qquad (3\,16)$$

where

k is the midpoint knot parameter,

t is the scaling constant,

a, **b** denote the unit vectors to the direction of the vectors (P_3-P_2) and

(P_4-P_3) respectively

The midpoint knot parameter *k* is constrained by

$$\left|\frac{P_3-P_2}{P_4-P_3}\right|_{\text{type III}} = \left(\frac{1-k}{k}\right)^3 \tag{3.17}$$

Compare this equation with the constraints in Type I and Type II splines where

$$\left|\frac{P_3-P_2}{P_4-P_3}\right|_{\text{type I}} = \left(\frac{1-k}{k}\right)^2$$

$$\left|\frac{P_3-P_2}{P_4-P_3}\right|_{\text{type II}} = \left(\frac{1-k}{k}\right) \tag{3.18}$$

The difference in the chord length ratio affects midpoint knot parameter *k* such that

$$k = \frac{1}{1+\left(\left|\frac{P_4-P_3}{P_3-P_2}\right|\right)^{\frac{1}{d}}}$$

where

d = 3 for Type III splines,

d = 2 for Type I splines,

d = 1 for Type II splines. (3.19)

Note that by varying the *d* value, a continuous set of splines can be produced The same holds true for the midpoint knot parameter *k* For instance, if *d* is equal to 2.5, then the characteristic of the spline will be somewhere in the middle of Type I and Type II splines Aside from the three types of G^1 splines, an almost unlimited number of splines can be produced Therefore, we define this constant as the *characteristic constant of the G^1 splines*

As a matter of fact, we have a single degree of freedom in the generation of the curve described by Equation 3.19 As long as we wish to incorporate the chord length information to the shape of a curve the characteristic constant *d* will determine the *k* value, and vice versa

Depending upon the design purpose, we can obtain a variety of curves which are visually continuous. Provided with a proper control of the characteristic constant, the curve can be made to vibrate or to flatten while preserving visual continuity An example using the characteristic constant, the control of motion path in an interactive design environment, will be developed later.

Subsequent evaluation of the left and right tangents yields the component ratio of

$$\frac{\text{a component}}{\text{b component}} = k^{m}(1-k)^{n}$$

where

$$\begin{aligned}(m,n) &= (1,\ -1) \text{ for Type III splines}\\ &= (0,\ \ 0) \text{ for Type I splines}\\ &= (-1,\ 1) \text{ for Type II splines} \qquad (3.20)\end{aligned}$$

The magnitude ratio of the left tangent to the right tangent becomes

$$\left|\frac{D_L}{D_R}\right|_{\text{Type III}} = \left|\frac{D_L}{D_R}\right|_{\text{Type II}} = \left|\frac{D_L}{D_R}\right|_{\text{Type I}} = \frac{k}{1-k} \qquad (3.21)$$

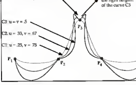

Figure 3-7 The continuity effect of the G^0 free form splines. By controlling the u, v parameters such that their sum becomes one, a motion path enabling such visual effects as bouncing balls can be achieved.

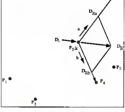

Figure 3-8. Decomposition of the slope vectors a, b denote the unit vectors to the direction of the vectors (P_3-P_2) and (P_4-P_3), respectively D_{Ra} denotes the tangential component of the right tangent, to the direction of the a vector Similarly for D_{Rb} In G^1 class splines, the magnitudes of the left tangent D_L and the right tangent D_R can be different, but the direction must remain the same

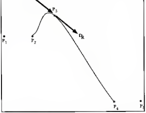

Figure 3-9 The tangents of an ordinary cardinal spline. Left and right tangents have the same magnitude and direction.

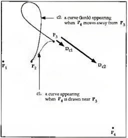
c2: a curve (kink) appearing when P_4 moves away from P_3
P_3
D_{c1}
P_1
P_2
D_{c2}
c1: a curve appearing when P_4 is drawn near P_3
P_4

Figure 3-10. Tangent constraints that cause the wiggles in an ordinary cardinal spline. D_{c1} and D_{c2} represent the tangents associated with the splines C1 and C2, respectively.

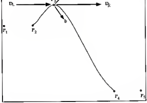

Figure 3-11. The tangents of a Type I G^1 free form spline: a, b denote the unit vectors to the direction of the vectors (P_3-P_2) and (P_4-P_3), respectively.

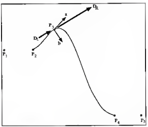

Figure 3-12. The tangents of a Type II G^1 free form spline: **a**, **b** denote the unit vectors to the direction of the vectors (P_3-P_2) and (P_4-P_3), respectively. The ratio of the magnitudes of the **a** component to the **b** component is 3.5 : 1 for this example

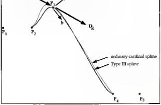

Figure 3-13 Comparison of a Type III G^1 free form spline with an ordinary cardinal spline a, b denote the unit vectors to the direction of the vectors (P_3-P_2) and (P_4-P_3), respectively The magnitude ratio of a component to b component is 1 : 3.5 for this example.

Experiments on the Removal of Wiggles

In the previous section, three types of G^1 free form splines were developed by imposing chord length dependencies on the determination of the midpoint knot parameters. In this section, we compare the visual properties of the three types of free form splines in terms of resolving the wiggle problem, which was the main motivation of this chapter.

A wiggle occurs when the distance between two neighboring alpha carbon atoms is small compared with the subsequent distances. The variation of the distances appears in the form of tangent constraints at data points. In this respect, all three types of free form splines are eligible for the interpolation of the backbone atoms, since they were developed in such a way that the tangent constraints of ordinary cardinal splines can be alleviated.

However, each type of free form spline has its own characteristic visual performance. In Figure 3-14, the ordinary cardinal spline method was used to interpolate five data points, P_1 through P_5. As can be readily seen, the curve shows a wiggle in the interval $[P_3,P_4]$, which is a relatively short interval compared with the interval $[P_2,P_3]$.

Figures 3-15 through 3-17 show the interpolation of the same data points using Type II, Type I, Type III free form splines, respectively. All of them contribute to eliminate the wiggle in the interval $[P_3,P_4]$, thus making them adequate for the interpolation scheme for the backbone curve.

Figure 3-18 shows a closer view of the behavior of the three types of free form splines in the region $[P_3,P_4]$. The Type II spline has the lowest level of overshoot in the region, while Type I and Type III show an increased level of overshoot. However, the converse is true for the longer interval $[P_2,P_3]$.

Figure 3-19 illustrates the visual properties of the curves starting from point P_2 up to about half way to point P_3. Type I and Type III splines almost parallel the curve of the ordinary cardinal spline, while the Type II spline exhibits a considerable amount of overshoot.

Therefore, we can conclude that the level of overshoot in the shorter interval is proportional to the value of characteristic constant d in Equation 3.19, while that of the longer interval is inversely proportional to the characteristic constant.

The decision as to which type of free form spline is better depends on the various design purposes. It is a subjective matter. A designer may regard the swing in the longer interval as a more natural one, or he might prefer a straight line in the longer interval and trade off the swing with that of the shorter interval.

Each type of free form spline can be extended to a surface interpolation scheme. For instance, the bilinear blending method can easily convert a combination of space curves into three-dimensional surface shapes. Figure 3-21 and Figure 3-22 show Coons' patch [Coo74, For72] applied to different interpolating schemes. The surfaces show the same trend as was seen in the case of a space curve. The surface blended by a cardinal spline tends to be bent at the small patch on the left side, while the surface produced by a Type II spline shows more inflection at the large patch on the right side. Side views (see Figures 3-23 and 3-24) demonstrate this property more clearly.

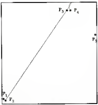

Figure 3-14. A wiggle produced by an ordinary cardinal spline.

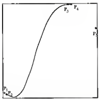

Figure 3-15. Removal of the wiggle by a Type II G^1 free form spline

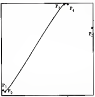

Figure 3-16 Removal of the wiggle by a Type I G^3 free form spline.

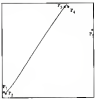

Figure 3-17 Removal of the wiggle by a Type III G^3 free form spline.

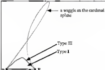

Figure 3-18. Comparison of free form splines in the shorter interval $[P_3,P_4]$.

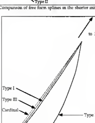

Figure 3-19 Comparison of free form splines in the longer interval $[P_2,P_3]$

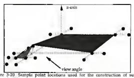

Figure 3-20 Sample point locations used for the construction of surface patches in Figures 3-21 through 3-24.

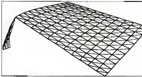

Figure 3-21 Coons' patch produced by an ordinary cardinal spline

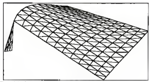

Figure 3-22. Coons' patch produced by a Type II spline

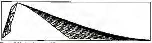

Figure 3-23. A side view of Figure 3-21

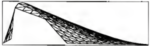

Figure 3-24. A side view of Figure 3-22.

Comparison with Tensioned Cardinal Splines

Having accomplished the task of removal of wiggles in the last section, we devote this section to the comparison of the properties of the G^1 free form splines with the well-known tensioned cardinal splines [Smi83]. In this section, the Hermite form of our free form splines will be compared with that of the tensioned cardinal splines. To distinguish free form splines from the tensioned cardinal splines, the equation for tensioned cardinal splines will be derived in the context of chord length parametrization. This way, the difference between the tensioned cardinal splines and our free form splines making use of the chord length information can be clarified. The visual properties of the two approaches will be explained in terms of removal and addition of tension.

A cubic spline can be expressed in terms of directional vectors instead of positional vectors (the location of data points). By a simple manipulation, the point locations can be converted into directional vectors. For instance, tha general matrix expression of a cubic spline is

$$\mathbf{P}(s) = [s^3\ s^2\ s\ 1]\ \mathbf{A}\ [\mathbf{P}_1\ \ \mathbf{P}_2\ \ \mathbf{P}_3\ \ \mathbf{P}_4]^T$$

Instead of writing the point matrix as $[\mathbf{P}_1\ \mathbf{P}_2\ \mathbf{P}_3\ \mathbf{P}_4]^T$, one can rewrite it in terms of directional vectors. In this method, vectors $(\mathbf{P}_3\text{-}\mathbf{P}_1)$ and $(\mathbf{P}_4\text{-}\mathbf{P}_2)$ are introduced in place of points $\mathbf{P}_1$ and $\mathbf{P}_4$ so that the point matrix becomes

$[\mathbf{P}_2\ \ \mathbf{P}_3\ \ (\mathbf{P}_3\text{-}\mathbf{P}_1)\ \ (\mathbf{P}_4\text{-}\mathbf{P}_2)]^T$

Consequently, the point matrix is made up of endpoints $\mathbf{P}_2$ and $\mathbf{P}_4$ at both ends of the curve, and two vectors $(\mathbf{P}_3\text{-}\mathbf{P}_1)$ end $(\mathbf{P}_4\text{-}\mathbf{P}_2)$ representing the slopes at $\mathbf{P}_2$ and $\mathbf{P}_4$, respectively. As a matter of fact, general Hermite interpolation allows for slope vectors other than $(\mathbf{P}_3\text{-}\mathbf{P}_1)$ and $(\mathbf{P}_4\text{-}\mathbf{P}_2)$, and thus representation belongs to a special case of the general Hermite interpolation expression. Using the directional vectors, the free form spline in Hermite form can be written as

$$p(s) = [s^3 \; s^2 \; s \; 1] \, \mathbf{A} \, [\mathbf{P}_2 \;\; \mathbf{P}_3 \;\; (\mathbf{P}_3\text{-}\mathbf{P}_1) \;\; (\mathbf{P}_4\text{-}\mathbf{P}_2)]^T$$

with

$$\mathbf{A}' = \begin{bmatrix} \frac{1}{u}-2+\frac{1}{1-v} & -\frac{1}{u}+2-\frac{1}{1-v} & u-2+\frac{1}{u} & -(1-v)+\frac{1}{1-v} \\ -\frac{2}{u}+3-\frac{1}{1-v} & \frac{2}{u}-3+\frac{1}{1-v} & -2u+4+\frac{-2}{u} & (1+v)-\frac{1}{1-v} \\ \frac{1}{u}-2 & 0 & u & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad (3.22)$$

Notice that most of the elements in each column of matrix **A** are functions of the u and v parameters. Thus the values of the matrix elements will reflect the change of the midpoint knot parameters u and v, if the parameters are associated with the chord length between the data points. However, as will be explained next, only the last two columns of the matrix **A**' will be dependent on chord length if the tensioned cardinal splines are expressed in Hermite form.

The use of tensioned cardinal splines has been a conventional method of adjusting curve shapes, using a parameter called tension, while preserving the C^1 continuity. Ordinary cardinal splines can be expressed in terms of the Bézier control points [Far90] which make use of the knot parameters to meet the tangent continuities at joints. As a result, the cardinal spline can be written as a function of knot parameters. An additional constraint that the knot spacing be proportional to the chord length leads to the tensioned cardinal splines

In Figure 3-25, data points $\mathbf{P}_1$ through $\mathbf{P}_4$ are to be approximated by a tensioned cardinal spline, and they are assumed to be data points corresponding to the parameter u which takes the values of u_1 through u_4. The tensioned cardinal spline can be derived starting with the construction of Bézier control points. If points $\mathbf{P}_2$ and $\mathbf{P}_3$ correspond to Bézier control points $\mathbf{b}_6$ and $\mathbf{b}_9$ respectively, then points $\mathbf{b}_7$ and $\mathbf{b}_8$ are are called the *inner Bézier control points* produced in relation to the other control points $\mathbf{b}_6$ and $\mathbf{b}_9$, respectively. The position of these control points can be evaluated by defining the knot space Δ_k as

$$\Delta_i = u_{i+1}-u_i$$

Using a method for tangent estimation known as FMILL [Far90], two inner Bézier points can be expressed as

$$\mathbf{b}_7 = \mathbf{b}_6+\frac{\Delta_2}{3(\Delta_1+\Delta_2)}\mathbf{l}_2$$

$$\mathbf{b}_8 = \mathbf{b}_9-\frac{\Delta_2}{3(\Delta_2+\Delta_3)}\mathbf{l}_3$$

where

$$\mathbf{l}_2 = \frac{\mathbf{P}_3-\mathbf{P}_1}{||\mathbf{P}_3-\mathbf{P}_1||}$$

$$\mathbf{l}_3 = \frac{\mathbf{P}_4-\mathbf{P}_2}{||\mathbf{P}_4-\mathbf{P}_2||}$$

with

$||\mathbf{a}||$ standing for the norm of vector **a**.

Using these values, the general Hermite expression of a cardinal spline in the interval $[\mathbf{P}_2,\mathbf{P}_3]$ becomes,

$$\mathbf{P}(s) = \mathbf{P}_2\, H_0^3(s)+\mathbf{P}_3\, H_3^3(s)$$

$$+\frac{\Delta_2}{3(\Delta_1+\Delta_2)} I_2 (3s^3-6s^2+3s)+\frac{\Delta_2}{3(\Delta_2+\Delta_3)} I_3 (-3s^3+3s^2)$$

$$= \left[s^3 \ s^2 \ s \ 1\right] \mathbf{B} \ [\mathbf{P}_2 \ \ \mathbf{P}_3 \ \ (\mathbf{P}_3-\mathbf{P}_1) \ \ (\mathbf{P}_4-\mathbf{P}_2)]^T$$

with

$$\mathbf{B} = \begin{bmatrix} 2 & -2 & \frac{\Delta_2}{||\mathbf{P}_3-\mathbf{P}_1||(\Delta_1+\Delta_2)} & \frac{\Delta_2}{||\mathbf{P}_4-\mathbf{P}_2||(\Delta_2+\Delta_3)} \\ -3 & 3 & \frac{-2\Delta_2}{||\mathbf{P}_3-\mathbf{P}_1||(\Delta_1+\Delta_2)} & \frac{-\Delta_2}{||\mathbf{P}_4-\mathbf{P}_2||(\Delta_2+\Delta_3)} \\ 0 & 0 & \frac{\Delta_2}{||\mathbf{P}_3-\mathbf{P}_1||(\Delta_1+\Delta_2)} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad (3\,23)$$

Notice that the first two columns of matrix **B** are constants in contrast with the Hermite form of the free form splines. The ordinary (in the sense that it is not tensioned) cardinal spline represents one instance of this expression where

$$\Delta_1 = \Delta_2 = \Delta_3 = 1$$

$$||\mathbf{P}_3-\mathbf{P}_1|| = ||\mathbf{P}_4-\mathbf{P}_2|| = 1.$$

The knot space variables Δ_1 and Δ_2 can be *parametrized by chord length* such that

$$\frac{\Delta_1}{\Delta_2} = \frac{||\mathbf{P}_2-\mathbf{P}_1||}{||\mathbf{P}_3-\mathbf{P}_2||}$$

$$\frac{\Delta_2}{\Delta_3} = \frac{||\mathbf{P}_3-\mathbf{P}_2||}{||\mathbf{P}_4-\mathbf{P}_3||}$$

This substitution, called "chord length parametrization," yields a C^1 continuous cardinal spline so that the derivative with respect to the parameter u is continuous

We can prove that chord length parametrization is C^1 continuous in the following way Let

$$a = \frac{\Delta_2}{||\mathbf{P}_3-\mathbf{P}_1||\ (\Delta_1+\Delta_2)}\ , \text{and}$$

$$b = \frac{\Delta_2}{||\mathbf{P}_4-\mathbf{P}_2||\ (\Delta_2+\Delta_3)}$$

Then the above Hermite form of cardinal spline reduces to

$$\mathbf{P}(s) = [s^3\ s^2\ s\ 1]\ \mathbf{B}\ [\mathbf{P}_1\ \mathbf{P}_2\ \mathbf{P}_3\ \mathbf{P}_4]^T$$

with

$$\mathbf{B} = \begin{bmatrix} -a & 2-b & a-2 & b \\ 2a & b-3 & 3-2a & -b \\ -a & 0 & a & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.24}$$

which is exactly the conventional tensioned cardinal spline Notice that this matrix agrees with that of the free form spline, if both midpoint knot parameters are equal to $\frac{1}{2}$ Therefore, the left tangent at point $\mathbf{P}_2$ is

$$\left.\frac{d\mathbf{P}(s)}{ds}\right|_{s=1} = b\,(\mathbf{P}_4-\mathbf{P}_2),$$

and the right tangent evaluated by a curve $\mathbf{Q}(s)$ to the right of point $\mathbf{P}_2$ is,

$$\left.\frac{d\mathbf{Q}(s)}{ds}\right|_{s=0} = a\,(\mathbf{P}_4-\mathbf{P}_2) \tag{3.25}$$

In order to assure C^1 continuity at the joint $\mathbf{P}_2$, the value of a and b must be the same Thus is the reason why the tension parameter t replaces both a and b in the tensioned cardinal spline, thus removing the extra degree of freedom of varying the two independent knot spacings

Although the locality of a spline is determined by the number of changed curve segments with a modification of a data point position, the locality can also be described by the number of changed curve segments with a change of a parameter value If the above parameter "a" (= b = t) is to have a different value, then the same value of "a" should be applied to entire curve segments. Otherwise, the joint between two curve segments whose tension parameter differs, fails to be C^1 continuous Consequently, the same tension value is applied indiscriminately to all curve segments, regardless of the distance between the endpoints.

Figure 3-26 illustrates the use of a tensioned cardinal spline for the removal of a wiggle in the same example as the last section In the interval $[P_3,P_4]$, the wiggle is removed Nevertheless, the interpolation in the region $[P_2,P_3]$ is almost linear, since the same tension value used to alleviate the wiggle was applied in this region The tension cannot be selectively applied in each interval because of the C^1 constraint Moreover, the long interval length in $[P_2,P_3]$ acts to flatten the curve Besides being unnatural, such linearity prohibits even a small amount of inflection on the curve, which is one of the reasons why the cubic spline was employed

Figure 3-27 illustrates another aspect of tensioned cardinal splines with curves corresponding to two different tension values. A curve with a tension value of .5 coincides with an ordinary cardinal spline, which is not tensioned. In a curve with a tension of 1.5, the left and right tangents at point P_3 are denoted by D_-, D_+, respectively Suppose we are not satisfied with the linearity of the curve with a tension of .5 in the interval $[P_3,P_4]$ Then we might be increasing the degree of inflection in this region by setting the tension as 1.5

However, the problematic area of the curve at hand is in the region $[P_2,P_3]$ this time Because of the application of the same tension value in the short interval, the curve shows overshoot in this region. In addition, the constraint of the tangential magnitude that the left and right tangents must be the same makes the curve show greater curvature near point P_3 than near point P_2 Therefore, resulting curve shape is hardly acceptable as a reasonable interpolation

In contrast, the midpoint knot parameters of G^1 free form splines can vary for each data point A parameter value at a data point affects only two curve segments adjacent to the point so that the parameter can be applied locally Most importantly, distance information can be incorporated into the spline in such a way that could adjust tension in the longer or shorter interval For instance, while reducing the wiggle in the shorter interval, the Type II spline can introduce an arbitrary amount of inflection in the linear region as was shown previously in Figure 3-15. Therefore, G^1 free form splines can be said to have more flexibility compared with the tensioned cardinal spline

It is important to note that the random assignment of a and b values can produce G^1 continuity at some data points, as can be deduced from Equation 3.25, since the left tangent is a scalar multiple of the right tangent Nevertheless, these random values affect the curve shape between the intervals in such an unpredictable way as to be unacceptable as a proper interpolation scheme. The curve may exhibit extreme oscillation or inflection between the endpoints, and the major factor that causes this behavior is that the curve should meet the very restrictive constraint, that the tangential direction must be (P_4-P_2) Such uncontrollable curve behavior between the endpoints has left the tensioned cardinal splines with more restrictive C^1 continuity

Although the Hermite form of cardinal spline appears to embody the knot spacing parameter, it is different from our formulation of free form splines. The first two columns of the B matrix of the cardinal Hermite form in Equation 3.23 are constant, even in chord length parametrization The only way the knot spacing could affect the spline is by way of the last two columns of the B matrix Although a modification of these two columns subsequently affects the magnitude of the slope at points P_2 and P_3, namely (P_3-P_1) and (P_4-P_2), only the magnitude of these vectors can be changed as a result of the variation in the knot spacing

In contrast, free form splines do use entire columns of Equation 3 22 to influence the curve shape, so that the resulting curve is not merely a change of tangent magnitude, but the adjustment of the tangential directions. To make a curve be G^1 continuous, there should be a degree of freedom in selecting the direction of tangents at given data points, since the direction immediately affects the tension of a curve [Man75] Otherwise, the curve will undergo an unnatural oscillation between these data points to adjust itself to the fixed tangential directions at these points.

One approach to remove this restrictiveness in the tangential direction of the data points is presented by Kochanek [Koc84] Starting with the Hermite form of ordinary cardinal splines he has modified the slope component of the point matrix, in such a way that the tangent direction can be adjustable in our terms, Kochaneck decomposed the fixed tangential direction at point P_3 into two components.

$$P_4-P_2 \;=\; (P_4-P_3)+(P_3-P_2)$$

In an attempt to control the curve shape, these two vectors are weighted by certain parameters such that

$$P_4 P_2 = k_1 (P_4 - P_3) + k_2 (P_3 - P_2)$$

where the weighting factors k_1, k_2 are freely modified to control the tangent and, thus the shape of the curve. In a general expression, his method in Hermite form corresponds to

$$P(s) = [s^3\ s^2\ s\ 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ D_1 \\ D_2 \end{bmatrix}$$

with

$$D_i = \frac{1}{2} (k_1 (P_{i+1} - P_i) + k_2 (P_i - P_{i-1}))$$

Although he extended the curve scheme to the point thet the left and right tangents are not the same, our discussion in this section will be restricted to the identical tangent case. Further discussion will follow in the next section.

This method can be characterized by two observations. First, the curve is not generally G^1 continuous, since the left tangent is identical to the right tangent. Rather, it can be said to be C^1 continuous where the left and right tangents are the same. Therefore, the variety of the curve shapes extractable from the G^1 class could not be covered. Second, despite the fact that it is C^1 continuous, the formulation is not based on the chord length information. Therefore, the method works well when the data points are equidistant, but it demands iterative readjustment of the weighting factors if the data points are not equally spaced. In practice, situations when data points are equidistant are rare The difficulties involved in the iterative manipulation of the shape control parameters are mentioned in the next section

In summary, free form splines are more flexible than tensioned cardinal splines in that the direction of tangents can be freely adjusted. From a mathematical viewpoint, they belong to the class of G^1 splines encompassing more diverse curve shapes than C^1 splines, which are a special case of G^1 splines Moreover, the chord-length dependency of free form splines facilitates the manipulation of general random-spaced data points which are not necessarily equidistant The flexibility in selecting the tangential direction of free form splines will be explained further in the next section, in conjunction with a method to control the curve shape interactively

Application. An Interactive Motion Path Generator

This section extends the use of free form splines as an interactive animation tool to generate a motion path. Most frequently, animators design a motion path interactively, rather than resorting to the three types of automatic means described in previous sections

To date, research into control of curve shapes has mainly relied on the introduction of new parameters [Nie74, Bars83, Bars84, Fol86, Koc84] The designers had a rough idea of what the curve will look like when they change the parameters Nevertheless, the exact behavior of the curve could hardly be predicted and it was left to the designer to iteratively vary and fine tune the parameter values until they reach a satisfactory curve shape This happens because the parameter values are scalars which cannot directly interact with our visual perception, while the data points related to the parameters are geometric. For instance, one may question exactly how a beta spline will behave near a data point if the bias parameter at the point is changed from 5 to 7.

The greatest advantage of G^1 free form splines developed earlier lies in the fact that the specification of tangent vectors at data points can directly lead to the piecewise formulation of the splines. Exploiting this advantage, the only thing designers need to do is to draw tangential directions for individual data points. They can specify how the curve will proceed at the data points simply by drawing a vector at that point. It is the responsibility of G^1 free form splines to produce curves which exactly match these tangents. This way the interaction between computer and designer becomes easier and simpler because it is no longer necessary to predict the visual effect that the scalar parameter will cause.

This section reviews previous research done on the control of curve shapes and presents a new method enabling more flexible control of the curve shapes based on G^1 free form splines.

Relevant Works

Before investigating previous works, let us briefly examine the dual usage of a spline curve.

A spline can be used for two purposes in animation. Figure 3-28 illustrates this point. At time t = 1, a snapshot of four data points P_1, P_2, P_3, P_4 can be interpolated by using a spline to produce a space curve. Suppose the four points change their positions as a function of time, such that their locations appear as in Figure 3-28 at time t = 2 and t = 3. With respect to point P_1, three data points are given in the time axis. Interpolating the three locations for the same point P_1 corresponds to generating a locus of movement in the time axis for the data point, and a spline was used for the generation of motion path for individual data points in space Therefore, control of the shape of the motion path can directly apply to control of the shape in space

There have been efforts to control the shape of three-dimensional space curves in terms of bias and tension [Nie74, Bars83, Bars84, Fol86, Koc84] The term control of tension represents making a curve tighter or looser at data points [Nie74, Koc84] or on the curve segments between data points [Bars84, Fol86, Sch66] In general, a curve can be said to be highly tensioned if the curve shape tends to be linear between data points For instance, a Type II spline exhibits lowest tension relative to other splines, as was shown in Figure 3-19

The term bias has been used to represent the variation of tangential direction at data points The bias can be used to simulate the traditional animation effect of following through after an action, or anticipating a movement [Tho81, Las87]. Actions hardly come to a sudden and complete stop, but are generally carried past their termination point For example, a hand will follow through after throwing a ball This tendency to preserve the previous direction can be shown near point P_3 in Figure 3-31 The tangent at point P_3 resembles the vector (P_3-P_2) so that the direction of the curve persists even after it passes through point P_3

On the other hand, anticipation prepares for a movement It is a technique to catch the audience's eyes, to prepare them for the next movement and let them expect it before it actually occurs For example, the tangent at point P_3 in Figure 3-33 resembles the vector (P_4-P_3) rather than the vector (P_3-P_2) Therefore, the motion path can be controlled to change its direction before it actually reaches point F_3, and prepares for the next movement.

As approximating splines, Beta splines [Bars83, Bars88] were modeled with the inherent capability of controlling the shapes using a pair of scalar parameters, bias and tension. Curve shapes are determined by the specification of two independent parameters. With a fixed bias value, the effect of the variation of tension can be easily recognized. Similarly, a bias effect can be examined only if the tension is set fixed. However, it is hard to predict a curve shape produced by the combination of the two parameters, since a tension between data points can affect the bias on the same data points. In practice, the two parameters interact with each other during the formulation of the curves, even if designers think they are independent.

In the area of interpolating splines, Nielson developed the v-spline [Nie74, Far90] as a polynomial alternative to Schweikert's exponential-based cubic spline [Sch66], the *spline under tension*. He noticed that the exponential function is a major impediment to efficient evaluation of the splines and replaced it with a G^2 continuous [Bars88] polynomial, where the difference of the second-order left and right derivatives is set as a multiple of the first-order derivative at a data point. The proportionality constant, being set as the tension parameter at the point, linearizes the curve as it approaches infinity. Unfortunately, the method lacks a mathematical proof of the existence of a unique solution [Bars83, Far90]. Experiments on negative tension values in the v-splines can be found in [Far85]. However, finding a good tension value is not a trivial matter and no automatic method for doing so is yet known.

Given data points, and the corresponding tangents, Hermite interpolation [Far90] can be used to produce C^1 cubic splines. The tangents should be specified in both magnitude and direction. The tangential direction can be predicted in such a way that it could fit the desired shapes. However, from a designer's viewpoint, the specification of tangential magnitude belongs to a difficult task. For instance, one might have no idea how the curve will change if the magnitude of a data point is multiplied by a factor of ten. Moreover, the curve generated by such repetitive manipulation belongs to the C^1 class, which has highly restrictive tangential constraints, and has less variety than the G^1 class free form splines.

Kochaneck [Koc84] has defined both bias and tension as a weighting function of two vectors formed by subtracting three adjacent data points. He even allowed the left tangent to be different from the right tangent at data points to define a continuity parameter. The subsequent curve shape is defined by the multiplication of the three parameters—bias, tension and continuity. One of the difficulties with this approach is that the combination of parameters may yield unexpected shapes. For instance, specification of zero continuity yields the same curve as a curve with a tension value of one. If an animator decides to let the tension be one, he should strive to make the continuity parameter approach zero to maintain a consistency in his specification of parameters. That is, he needs to have an a priori knowledge of what parameters conflict with one another. Another disadvantage of this approach is that the curves in general are not only discontinuous in tangential magnitude, but also discontinuous in the direction of tangents. Generally, the curves are neither C^1 continuous nor G^1 continuous.

To characterize previous research done in the control of curve shapes, it can be said that all of the methods incorporate too many burdens to be useful to designers. For instance, one may randomly specify the values of the three parameters—bias, tension and continuity. The animator should be prepared with some knowledge of what bias, tension, and continuity are

Then he may increase and decrease these values based on that knowledge. Sometimes he may encounter unexpected curves caused by interference between the parameters, and have to replace the parameter values with what he may think would have a high probability of yielding the desired curve.

Tangent Specification Method

From an animator's point of view, the previous method demands too much imagination. Those methods require that the operator supply *scalar* values of parameters and these scalar values could not appeal to the visual prediction of the curve shapes. Imagination must be mobilized to predict curve shapes from given scalar parameters. When three parameters are involved, such visualization is not easy, even though the concept of individual parameters and the possible interactions among them are kept in mind.

General G^1 free form splines allow us to reverse the above procedural direction. That is, an imaginary curve will constrain the parameter values. Imagine a curve interpolating given data points without any concern or knowledge of the concept of the shape control parameters. Then one can specify the tangential directions by drawing a line segment starting from individual data points. These tangents in turn will be used in the formulation of the piecewise G^1 free form splines. In this method, it is the programmer's responsibility to convert the tangent information into corresponding *k* parameters of the G^1 free form splines. Therefore, the only task left to the animator is to draw tangential directions based on his imagination. It is important to note that shape control is now done by visual vector drawing instead of the specification of numbers.

Figure 3-29 illustrates initial vector specification. Points P_1 through P_6 represent the data points to be interpolated and we limit our interest to curve behavior near point P_3 and P_4 for illustrative purposes. Animators can imagine the desired curve and draw tangent vectors at some data points, which for this example are points P_3 and P_4. The magnitude of the vector does not matter, since only the directional information is used in the production of G^1 free form splines. In the figure, the direction of the tangent at point P_3 resembles the vector $(P_3$-$P_2)$ more closely than the vector $(P_4$-$P_3)$, so that the effect of follow-through can be simulated. Similarly the tangential direction of point P_4 resembles the vector $(P_4$-$P_3)$. Therefore the bias is effectively expressed by the specification of tangent vectors starting from the data points.

Figure 3-33 shows another instance of vector specification that will lead to increased tension in the interval $[P_3, P_4]$. Both tangential directions at point P_3 and P_4 are similar to the vector $(P_4$-$P_3)$. As the tangents approach these directions, the curve will tend to become linear.

In essence, G^1 free form splines do not distinguish between the bias and tension parameters. The direction of tangents determines both. Tension is treated as another aspect of bias appearing when the direction of bias is modified.

Figure 3-30 illustrates a procedure required to compute midpoint knot parameter k, based upon the tangent specification shown in Figure 3-29. The vectors D_3 and D_4 represent the tangents at point P_3 and P_4, respectively. The tangent at P_3 is decomposed into its component vectors $(P_3$-$P_2)$ and $(P_4$-$P_3)$. The magnitudes of these incoming and outgoing vectors are marked a and b, respectively and they can be calculated by a simple dot product of two vectors. For instance, the magnitudes of the component vectors at point P_3 are:

$$a = (P_3 - P_2) \cdot D_3$$
$$b = (P_4 - P_3) \cdot D_3 \qquad (3.26)$$

The same holds true for the tangent $\mathbf{D_4}$ at point P_4 except that the incoming and outgoing vectors are replaced by $(P_4 - P_3)$ and $(P_5 - P_4)$, respectively. A direct consequence of Equation 3.9 is that the ratio of these two magnitudes is a function of midpoint knot parameter *k* and the length of the incoming and outgoing vectors.

$$\frac{a}{b} = \frac{(1-k)^2}{k^2} \frac{|P_3 - P_2|}{|P_4 P_3|} \qquad (3.27)$$

Note that this expression is valid both for the left and right tangents with different magnitude. The value of the midpoint knot parameter *k* is then

$$k = \frac{1}{1 + \sqrt{\frac{a}{b} \left| \frac{P_4 - P_3}{P_3 - P_2} \right|}}$$

(3.28)

Therefore, a unique midpoint knot parameter value is determined given the direction of a tangent. The range of *k* is confined to $(0 < k \leq 1)$ by the above equation. In addition, since the *k* value of one is not defined by Equation 3.3 and 3.9, such a situation can be avoided by assigning an infinitesimally small value, in case a component is zero, which happens when the tangent is perpendicular to the vector $(P_3 - P_2)$. In the special case of

$$\frac{a}{b} = 1,$$

Equation 3.28 reduces to a Type I spline as expected. As can be seen Figure 3-31, the resulting curve is G^1 continuous while being faithful to its original tangent specification given in Figure 3-29.

Note that the decomposition of vectors and thus the calculation of midpoint knot value k can be handled completely by an algorithm so that animators do not have to be concerned about it

Figure 3-32 illustrates another aspect of the tangent specification method from the point of view of tension control. In this figure, the relaxation of tension appears as two extraneous inflection points, inflection point 1 and inflection point 2 in the region $[P_3, P_4]$ Compare this with the curve in Figure 3-31 which shows a single reflection point after point P_3 The inflection point 2 is caused by making the tangent at point P_4 similar to the vector $(P_5\text{-}P_4)$ rather than $(P_4\text{-}P_3)$ The curve shows inflection before it reaches point P_4 to meet the requirement of the tangent direction at that point Therefore, the tension can be controlled near both ends of the interval $[P_3, P_4]$, either individually or simultaneously Often, preference is given to tighter curves to prevent an overshoot Figure 3-33 illustrates another case of the tangent specification with corresponding curve shape In this figure, both tangents at points P_3 and P_4 are leaning toward the vector $(P_4\text{-}P_3)$, making the curve linearized in the interval $[P_3, P_4]$ The more each tangent resembles the vector, the higher the curve will be tensioned

In G^1 free form splines, a midpoint knot parameter divides the curve segment into two intervals centered at that point Therefore, a cubic curve in one interval is affected by two midpoint knot parameters corresponding to the two endpoints at both ends of the interval In addition, two exterior points neighboring them also affect the curve segment. Hence modification of a data position affects four curve segments including the curves not neighboring the point Compared with this, modification of a tangent at a point affects only two curve segments; the segments to the left and right of the point, whose visual properties upon change of the tangent direction can be predicted intuitively. Therefore, the tangent specification method facilitates tightly localized control of curve shape without further complicating nearby curve segments.

Figure 3-34 illustrates shape control by a general free form spline which is not G^1 continuous. For instance, an animator may want to make the curve in the interval $[\mathbf{P}_3,\mathbf{P}_4]$ highly tensioned, yet may not be satisfied with the inflection in the interval $[\mathbf{P}_2,\mathbf{P}_3]$ appearing in Figure 3-33. Since the inflection to the left of point $\mathbf{P}_3$ in Figure 3-33 is caused by the G^1 constraints, one now has to sacrifice this continuity to remove the inflection. The G^1 continuity has to be traded off for more freedom in controlling the shape. In this figure, $\mathbf{D}_{L3}$ and $\mathbf{D}_{R3}$ represent the left and right tangents at point $\mathbf{P}_3$, while $\mathbf{D}_{L4}$ and $\mathbf{D}_{R4}$ are those at point $\mathbf{P}_4$. The tangents $\mathbf{D}_{L3}$ and $\mathbf{D}_{R4}$ are close to the vectors $(\mathbf{P}_3\text{-}\mathbf{P}_2)$ and $(\mathbf{P}_5\text{-}\mathbf{P}_4)$ to make the curve highly tensioned in the interval $(\mathbf{P}_3\text{-}\mathbf{P}_2)$ and $(\mathbf{P}_5\text{-}\mathbf{P}_4)$. The bias caused by tangents $\mathbf{D}_{R3}$ and $\mathbf{D}_{L4}$ shows a relaxed tension in the region $[\mathbf{P}_4\text{-}\mathbf{P}_3]$. This dual definition of the tangents agrees with our original definition of free form splines, where the parameters u and v were used.

For instance, the parameter u can be replaced by the k value matching the tangent $\mathbf{D}_{R3}$, and the parameter v can be replaced by the k value matching the tangent $\mathbf{D}_{L4}$ to produce the curve in the interval $[\mathbf{P}_3,\mathbf{P}_4]$. The midpoint knot parameter k can be calculated using the same decomposition procedure as described above, and inserted into Equation 3.3 to yield a general expression for free form splines. Comparison of the curve near points $\mathbf{P}_3$ and $\mathbf{P}_4$ shows that the curve near point $\mathbf{P}_3$ has more proximity to the G^1 free form spline than the one near point $\mathbf{P}_4$. If the left and right tangents are more alike in their direction, the generated curve exhibits more visual continuity near that data point. Exploiting this observation, we can even control the degree of discontinuity by adjusting the direction of the left and right tangents at individual data points

In summary, we have developed an intuitive method for control of the motion path during animation. Compared with the previous approach that is purely number oriented, our tangent specification method leads directly to formulation of piecewise curves that match the specified tangent directions An intentional violation of the visual continuity can also be incorporated into this method where the degree of discontinuity can be controlled by the similarity between the left and right tangents Moreover, bias, tension, and continuity parameters are integrated into the directional tangents in our method

Summary

By noting the characteristics of the visually continuous class of splines, we could remove the undesirable feature of the general interpolating splines, called wiggles

Our contribution in this chapter is that the visually continuous class of splines has been mathematically formulated in an analytic matrix form To date, no known solution to the visually continuous class of splines has been presented in its analytic form. By extending the class of splines with plain vector calculus, we could define characteristic constant d that could be effectively utilized to control the curve shapes, and to remove the wiggles as well Finally, as an application of the use of the characteristic constant an interactive design tool for motion path generation was developed

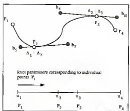

Figure 3-25 Ordinary cardinal spline in terms of the Bézier control points. The knot parameter u domain is mapped into the entire curve in the region [P1,P4]

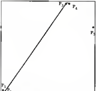

Figure 3-26 A tensioned cardinal spline used to remove a wiggle. The tension value is .05

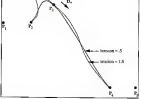

Figure 3-27 Slope vectors of tensioned cardinal splines D_- and D_+ represent the left and right tangents at point P_3, respectively.

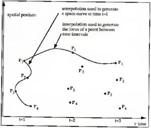
spatial position
interpolation used to generate a space curve at time t=1
interpolation used to generate the locus of a point between time intervals
P_1
P_1
P_1
P_2
P_2
P_2
P_3
P_3
P_3
P_4
P_4
P_4
t=1
t=2
t=3
t time

Figure 3-28 A spline used for two purposes—for the interpolation of data points in a static scene, and for the interpolation of a locus of a data point in the time domain

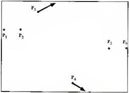

Figure 3-29 Direction of tangents specified by an animator

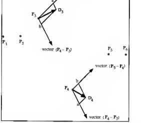

Figure 3-30 Construction of midpoint knot parameters by a decomposition of the directional vectors in Figure 3-29 Computed k values are 5, 29, .5, .25 at points P_2 through P_5 sequentially

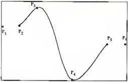

Figure 3-31. Final G^1 free form curve shape matching the tangent specification in Figure 3-29

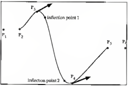

Figure 3-32. Reduction of tension in both the right of point P_3 and the left of point P_4. Computed k values are .5, .25, .66, .5 at points P_2 through P_5 sequentially

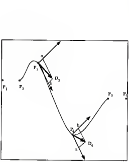

Figure 3-33 Increase of the tension in the interval [P_3,P_4] D_3 and D_4 represent the tangents at points P_3 and P_4, respectively Computed *k* values are .5, .5, .5, 5 at points P_2 through P_5 sequentially

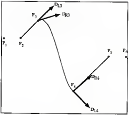

Figure 3-34 Breakup of G^1 continuity $\mathbf{D}_{L3}$ and $\mathbf{D}_{R3}$ are the left and right tangents at point $\mathbf{P}_3$, while $\mathbf{D}_{L4}$ and $\mathbf{D}_{R4}$ are those at point $\mathbf{P}_4$ Computed k values are 1 for $\mathbf{D}_{L3}$, 25 for D_{R3}, .5 for $\mathbf{D}_{L4}$, 1 for D_{R4} Additional tangent condition is that the right tangent of point $\mathbf{P}_2$ has k value of 1 and the left tangent of point $\mathbf{P}_5$ has that of 1.

CHAPTER 4
CONTROL OF MOTION SPEED IN ANIMATION

Introduction

The display of animation sequences is produced by showing a series of changing still pictures at a constant repetition rate. For video-compatible systems, the frame rate is 30 per second to create the illusion of reasonably smooth motion. In particular, the speed of animation is controlled by the number of frames for a given sequence. For instance, consider the animation of an atom moving through a pair of fixed positions, A and B in space. If the motion between the two positions is captured and displayed with 100 frames this motion will appear about ten times slower than the one with ten frames. Replay during sports broadcasting applies this visual effect for closer inspection of movement.

If the motion path of the atom is expressed in terms of a normalized parametric curve P(s), then it can be said that the atom moves from A to B while the parameter s runs from zero to one. That is, the locus of the atom is a mapping from the s parameter domain into three-dimensional space. In case eleven frames are required to describe the motion from A to B inclusively, the parameter s may be divided by ten, such that the first frame is generated by an s value of zero and the second frame is generated by an s value of .1, and the third by an s value of .2, and so on. Then the question is, "Does the atom exhibit constant motion speed with the frames generated by these parameter values ?" Theoretically, if the distance the ball travels between frames is identical through the entire sequence of frames, the motion speed is constant.

Unfortunately, constant spacing in the parameter domain does not guarantee constant spacing in the space domain except in the case of a linear motion path. This may be considered as a sampling problem in the parameter s space: in general, dense sampling expands the detail of the trajectory, providing a slow-motion version of the action, and sparse sampling compresses the detail, providing a time-lapse version of the action. Nevertheless this is not necessarily true for the motion path generated by a cubic spline where a small increment of the parameter s may produce a large distance gap and vice versa. A further requirement concerning the control of the motion speed is, "Can we even accelerate and decelerate the motion speed in the parametric curves by proper sampling of the s parameter?" This problem is known as *kinematic adjustment* or *control of motion speed*, or *motion dynamics* in animation, and thus chapter is devoted to solving this problem.

In an animation of protein folding, the motion speed of a single atom can be said to determine the speed of the transition from one folding state to another. Since our animation system for protein folding is based on the inbetweening of key frames representing the folding intermediates, the control of speed in the inbetweened frames is important. Figure 4-1 illustrates the case when no motion control strategy is taken. Consider the position of an alpha carbon at three sequential time instances (t = 0), (t = 1), and (t = 2). The alpha carbon in each instance can be assumed to be part of the whole backbone chain representing a folding state. Given the three folding states corresponding to each time instance, spline interpolation can be used to interpolate the motion path of the atom. However, the motion path is continuous while the atomic positions to be generated are discrete One cannot generate an infinite number of atomic positions Unfortunately, the splines employed to produce the motion path do not a offer convenient mechanism to place each atom with a proper interval between the atoms. If constant knot spacing is used in an attempt to make a constant inter-atomic distance, the atomic positions appear as in Figure 4-1. Note that the positions do not indicate multiple positions in a fixed time sense. Instead, the figure represents the positions of a single atom as time passes Consequently, when one displays these frames each made up of the given sequential atomic position the motion is explicitly slowed down near the position at time (t = 0), (t = 1) and (t = 2) In the meantime, the motion speeds up and then slows down in between The example shown here is a particular case and the relative spatial distribution of the inbetweened atoms is even less predictable in general As long as the unnecessary slowdown and speedup persists, and as long as there is no method to put the speed under control, one will meet with difficulty in capturing the smooth motion transition which is the original purpose of the animation.

Figure 4-2 illustrates the atomic positions produced by the speed control method described in this chapter Although it can also incorporate various effects such as acceleration and deceleration, the constant speed is applied in this example. As can be seen, the inter atomic distance is evened out to some degree, thereby assuring constant motion speed

This chapter develops a method to control the motion speed, using an approximation in a discrete parameter domain Two subsequent adjustment techniques, Lookahead Adjustment and Averaging Adjustment, will be combined with Incremental Knot Spacing to make up our motion control method

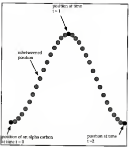

Figure 4-1 A spatial distribution of an atom in time domain with constant knot spacing Between the time (t = 0) and (t = 1), (t = 1) and (t = 2), the motion is accelerated and then decelerated upon sequential display of the atomic positions.

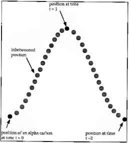

Figure 4-2 A spatial distribution of an atom in time domain when the speed control method described in this chapter is used to generate a constant speed.

Literature Review

Animation in the context of computer graphics can be classified in two categories. In *keyframe animation*, the motion is determined from a series of interpolatory positions between pairs of key frames while in *dynamic simulation*, the motion is determined from a physical law. Given a pair of key frames, generation of arbitrary interpolatory frames is often called *inbetweening* and the frames thus generated are called *inbetween frames*. Usually, the inbetweening is done between a pair of points belonging to the corresponding key frames, and the inbetween frames can be constructed from the collective intermediate points making up a still frame. Our concern in this chapter is with keyframe animation particularly in the control of timing in the automatic inbetweening system.

In most early keyframe animation, linear interpolation was used. For instance, in the animation system developed by Burtnyk and Wein [Bur71, Tha85], an example is given about the transformation of an alphabet to another alphabet character. If the character "n" is to be changed into the character "R", then both characters are decomposed into equal number of line segments so that an endpoint of the line segment in "n" can be mapped into another endpoint in "R" in a one-to-one manner. Between the two endpoints, the motion path is assumed to be linear. The primary reason the linear motion path is employed here is thet linear interpolation assures relatively easy control of motion speed. For instance, the motion speed will be constant with constant increase in parameter s domain. Nevertheless, linear interpolation cannot produce a smooth motion path and sometimes it even shows distortion of an object while in animation [Koc84]. Thus it was abandoned in favor of a more natural cubic spline interpolation.

In subsequent development, a real object was represented with a simplified form, a stick figure or a skeleton [Bur76]. Since a skeleton can be drawn easily, the dense skeletons drawn between key frames take the role of additional intermediate key frames so that the motion speed and the motion path can closely be controlled. Although Akima's parametric curve fitting [Aki70] is used in this scheme to smooth the motion path, the motion speed and motion path in this scheme are mainly controlled by an interactive trial and error procedure in the lowest level from which no further inbetweening will take place. Since the user is required to manipulate the inbetweened frames, the burden of kinematic adjustment is left to the user in practice. Further, the polygons enclosing the stick figures still were associated with linear artifacts [Bart89] in this method.

Another approach that requires a user specification down to the level of inbetween frames can be found in GENESYS [Bae69]. Here the animator provides a combined description of the path and timing of a motion by drawing motion paths called P-curves on a digitizing tablet. The shape of a P-curve defines the inbetweening trajectories of the motion path. In practice, a trail of symbols is used instead of a continuous line to depict the path and the symbols represent where the inbetween frames should be put on the motion path. In addition, the symbols are spaced equally in time so that the dynamics are represented by the local density of the symbols. Therefore, control of the motion speed, which is represented by relative spacing of the inbetween frames, should be dependent entirely on the manual interactions.

The moving point constraint in Reeves' approach [Ree81] offers a further degree of control to the P-curve method. A moving point is defined much like the symbols in a P-curve: a curve in space and time which constrains both the trajectory and dynamics (i.e., path and speed) of a point. While the P-curve is focused on the motion of a single point, this scheme allows the specification of the motion of multiple points. For instance, a curve as an object of animation can be approximated by multiple points lying on it. If an animator sketches the motion path for each of these points, the set of key frames and the set of moving points specify a patch network of the motion sequence. Then the set of key frames in a single patch is automatically inbetweened using such methods as in [Miu6, Coo74]. However, the motion speed in this method still had to be manually adjusted.

Steketee and Badler [Ste85] addressed the problem of the isolation of the temporal aspect from the spatial aspect of animation so that each can be put under separate control. The control of these aspects was made possible by the composition of two functions, the kinetic interpolant and the position interpolant. The kinetic Interpolant expresses the keyframe sequence as a function of time, and contains no information about the actual values of the motion parameter (i.e., knot parameter) that determines the position of the key frame. The position interpolant expresses the spatial position defined by the key frames, as a function of keyframe sequence. Therefore modification of the kinetic interpolant changes the timing of the key frames, and hence the speed and acceleration of the motion, without causing any change in the spatial position of the key frames. In other words, the motion speed was indirectly controlled by adjusting the time interval between the display of successive key frames, whose spatial position is fixed by the position Interpolant. Although the function for frame times was available for the animator to modify, the positional change induced by the modification of the frame time cannot be intuitively grasped with this approach.

Bartels and Hardtke [Bart89] allowed direct control of the motion speed by adjusting the sampling speed in the knot parameter domain. Their approach was aimed at providing the animator a means of a direct manipulation of the object-space speed rather than the indirect control of speed achieved through modification of frame times. In this approach, the desired magnitude of the speed (i.e., distance between the inbetween frames) on the parametric curve **F**(s), as a function of a motion speed parameter α, is defined as a *speed-profile curve*. On a parametric curve **F**(s), the knot parameter s can be related to α such that equally spaced values of α will produce values of s that lead to the desired sampling pattern (i.e., motion speed) of the inbetween frames on **F**(s). That is, the function relating the knot parameter s and the speed parameter α needs to be determined. If we define this function as the*knot profile*, then the problem of control of motion speed in their approach reduces to solving a differential equation about the knot profile defined in terms of the profile curve and the spline function adopted for the inbetweening.

However, there are a few problems associated with this approach. First, finding a sequence of the knot parameter s requires iterative evaluation of the Runge-Kutta integration [Pre88], which can be uneconomical in terms of the computational speed. Second, the root finding problem related to the boundary condition of the differential equation is difficult to handle in some cases, because it intrinsically has a tendency to diverge, depending on the situation [Pre88]. Third, this method does not necessarily result in "hitting the key frames." That is, the key frames may not be shown during animation. The effort to control the spatial distance between the inbetweened frames removes key frames from the animation stream instead. The main reason for the loss of the key frames stems from the cumulative errors associated with numerical evaluation of the differential equation.

The approach taken in this chapter shares some ideas with the above approach. For instance, we adopt the speed profile curve, and use a similar relation between the motion speed and the inter-frame distance. However, our approach removes the evaluation of the differential equation and replaces it with simpler calculations followed by algorithmic adjustments. The direct consequence of incorporating the algorithmic adjustment procedures is that it guarantees one will hit the key frames while providing appropriate motion speed to the inbetweened frames. The work done in this chapter is required, since our prototype system described in Chapter 5 is based on the assumption that the key frames must be part of the animation sequence.

Incremental Knot Spacing Method

The principal difference between Bartels and Hardtke's approach [Bar89] and our approach presented in this section lies in the method for the determination of the knot parameter s that controls the motion speed. While they evaluated the knot parameter s in a continuous fashion requiring Runge-Kutta integration, our calculation replaces it with a technique called incremental knot spacing. This technique, combined with a few adjustments described in subsequent sections, puts the control of motion speed on a simple calculation basis. This section describes a method to calculate the knot parameter s that could reflect the desired the motion speed in animation.

Although a rough concept of the speed profile curve can be found in [Bar89, Mas86, Ste85], it needs to be more precisely defined in our approach, it is the relative distance between the inbetweened frames as a function of frame sequence number

$$M_i = \phi\,(\alpha_i) \tag{4.1}$$

where

M_i is the relative inter-frame distance (i.e., speed),

α_i is the frame sequence number (i.e., $\alpha_i = 1, 2, ...$),

ϕ is the mapping that assigns a desired speed value for each frame number.

For instance, the value of M_i is constant if a constant speed is desired. If five inbetweened frames need to be produced with constant speed, the M_i values for each frame may be all 1's. However, assignment of all 10's still generates a constant speed in our approach, and thus the term relative distance instead of distance is employed in the definition of the speed profile curve. Since the absolute speed at a certain frame does not have any meaning in itself, and the speed relative to other frames affects the spatial positioning of the frame on the curve, the definition is valid without loss of generality. It should be noted thet this definition differs from the definition of speed in a common sense. Loosely speaking, speed in this context is the distance traveled per unit frame, while physics defines the term as the distance per unit time.

It is important to note that the conventional definition of the speed control perameter α is slightly different from ours. In the conventional definition, the values of α are assumed to be continuous and no explicit relation is assumed between the α values and the frame sequence number. In our approach, the α values can be only discrete integers (i.e., $\alpha_i = 1, 2, ...$) representing the frame sequence numbers. The reason for this discretization of α is connected to our subsequent approach to discretize the determination of the knot parameter. In addition, the speed profile curve becomes more intuitive since the profile represents the speed at the frames instead of the speed in relation to the motion parameter α. Figure 4-3 is an example of a speed profile curve. The speed values of the sequential inbetween frames numbered 1 through 5 are shown with corresponding magnitudes, ϕ_1 through ϕ_5. Note that what we define as a speed profile is the discretized version of the conventional continuous speed profile curve $\phi(\alpha)$. Hence the continuous profile can be approximated by the discreet profile simply by sampling the continuous speed profile curve with equidistantly spaced α values where the number of samples is equal to the number of inbetween frames.

Moreover, the speed values between the key frames can be interpolated such thet the inbetween frames can be assigned interpolated speed values. There is no restriction on the interpolation scheme in selecting the proper speed. The scheme to control the speed may be a second order polynomial to enable acceleration and deceleration. Figure 4-4 is an example of the use of a B-spline to interpolate the speed across several key frames. One advantage of interpolation of speed values is that it smooths and spreads out the speed values as it does in the interpolation of spatial data points. Whatever the continuous profile curve is, the discrete speed values can readily be taken by sampling. For instance, the ϕ_i in Figure 4-4 is the discrete speed value corresponding to the inbetween frames with sequence number i.

Given the desired speed profile along with the interpolation schema, the conventional approach is focused on finding a function thet relates these two input constraints to the determination of the knot parameter s

$$s = s(\alpha)$$

such that the equally spaced values of α will produce values of s spaced along some desired pattern of points on the trajectory $P(s)$. To distinguish this curve from the speed profile we use the term knot profile as a function from the keyframe sequence number α into the knot parameter s. Application of this function can be described as follows. Consider the case of constant speed in which the distance between the inbetween frames needs to be identical. In other words, we wish to maintain constant distance between the sample points on curve $P(s)$. According to the conventional approach, the parameter α is supposed to advance with constant increments. Then the knot parameter s will be evaluated for each of these α values using the above equation. Finally, each evaluated knot parameter s will be plugged into the curve representing $P(s)$ to generate the sample points with constant distance. Note that the knot parameter s does not increase with a constant increment even though the parameter α, and the final distance between frames does. Therefore, the problem of control of the motion speed is reduced to finding the function that could relate the constant α to the desired sampling pattern on $P(s)$.

Since, in the conventional approach, the knot parameter s is a continuous function of the motion speed parameter α, evaluation of $ds/d\alpha$ requires a numerical approximation where α values need to be evaluated far more densely (e.g., by around a factor of 10) than the number of inbetween frames, and still the result exhibits cumulative errors. The method is too costly, particularly when a local acceleration or deceleration needs to be roughly approximated.

Instead of using the above expression for the knot parameter s, Incremental Knot Spacing sets the incremental difference in the knot parameter s to be a function of the discretized speed profile ϕ_i with ($i = 1$ number of frames),

$$\Delta s_i = s_{i+1} - s_i = f(s_i, \phi_i) \tag{4.2}$$

The subscript i designates the $i^{\underline{th}}$ inbetween frame such that s_i produces the $i^{\underline{th}}$ inbetween frame upon insertion into the spatial interpolation curve F(s) If a knot parameter s_i is determined for one frame, the next knot parameter s_{i+1} is determined utilizing a function of the current knot value s_i and the speed profile value ϕ_i Note that by defining this way, the continuous relationship between the speed profile variable α and the knot parameter s can be avoided and hence numerical integration becomes unnecessary

The problem of the parametric representation of a curve lies in the fact that the incremental chord length is not constant with constant Δs In Figure 4-5 the distance represented by ΔL_i is the spatial distance between the position of the $i^{\underline{th}}$ frame at F_i and that of the $(i+1)^{\underline{th}}$ frame at F_{i+1} Both positions F_i and F_{i+1} are mapped from the knot parameters s_i and s_{i+1} into three-dimensional space and thus the increment in the knot parameter domain, Δs_i is mapped into the spatial displacement ΔL_i Even though the knot parameter s increases with constant steps (i e , $\Delta s_1 = \Delta s_2 = \quad = \Delta s_{n-1}$), the corresponding increment in the space, ΔL_i varies in general The deformity, that a uniform increase in parameter domain yields nonuniform spatial distribution in space domain, is typical of the behavior of general cubic splines where

$$F_i = f(s_i^3, s_i^2, s_i, 1)$$

$$F_{i+1} = f(s_{i+1}^3, s_{i+1}^2, s_{i+1}, 1)$$

The incremental chord length between any two adjacent points is

$$\Delta L_i = | P_{i+1} - P_i |$$

$$= \sqrt{\sum_{k=x,y,z} (P_{i+1,k} - P_{i+1,k})^2} \qquad (4.3)$$

where

the summation takes over the x, y, z components of the three-dimensional Euclidean space, and

$P_{i,k}$ is the k-component of the positional vector P_i

Therefore, evaluation of the distance between frames involves a square root of a sixth order polynomial about knot parameter s, whose coefficients depend on the interpolation scheme. Because of this complexity, it is difficult to predict the behavior of ΔL upon constant increase in the parameter s, Δs. Moreover, a relatively large increase in the parameter domain can produce a relatively small spatial displacement between frames. Therefore, the attempt to control the speed ΔL by adjusting Δs through the use of above expression fails.

In our method, we concentrate on the tangent information attainable from a given interpolation scheme. In differential geometry [Car76], the arc length of a parametrized curve P(s) with normalized parameter s is given by

$$L(s) = \int_0^1 \left| \frac{dP(s)}{ds} ds \right|,$$

with the tangent vector

$$\left| \frac{dP(s)}{ds} \right| = \sqrt{(\frac{dx(s)}{ds})^2 + (\frac{dy(s)}{ds})^2 + (\frac{dz(s)}{ds})^2} \qquad (4.4)$$

This is true for any parametrization including the parametrization by chord length. One consequence of Equation 4.4 is that the tangent vectors can be used as a measure of distance between adjacent inbetween frames since it is the rate of the change in arc length, or the chord length if the inbetween frames are densely spaced For instance, if the magnitude of the tangent vector at a certain frame on a curve is large, the position of the next frame is inclined to be farther apart spatially If the tangent vector at point P_i is denoted by $\mathbf{d}_i$,

$$\mathbf{d}_i = \left.\frac{dP(s)}{ds}\right|_{s=s_i}$$

with

$$|\mathbf{d}_i| = \sqrt{d_{ix}^2 + d_{iy}^2 + d_{iz}^2}$$

$$d_{ix} = \left.\frac{dx(s)}{ds}\right|_{s=s_i} \quad d_{iy} = \left.\frac{dy(s)}{ds}\right|_{s=s_i} \quad d_{iz} = \left.\frac{dz(s)}{ds}\right|_{s=s_i} \tag{4.5}$$

Figure 4-5 shows a tangent vector $\mathbf{d}_i$ at a point $\mathbf{P}_i$, a frame position in space produced by mapping from the parameter s_i As is the case with differential geometry, singular points leading to a $|\mathbf{d}_i|$ value of zero should be avoided in the evaluation of $\mathbf{d}_i$ Since the zero tangent means that the frames overlap at the same position, this situation does not occur in the normal interpolation process To further understand the properties of the tangent vectors in cubic splines, the magnitude of the vectors was examined with constant increase in the parameter s.

Figure 4-7 shows the position of inbetween frames based on the two-dimensional key positions in Figure 4-6 The region [key frame #2 key frame #3] and [key frame #3, key frame #4] of Figure 4-6 are interpolated using an ordinary cardinal spline Nine inbetween frames are generated between key frames #2 and #3 Figure 4-7 also illustrates the variation in the magnitude of tangent vector $\mathbf{d}_i$ with constant spacing of the knot parameter (i.e., $\Delta s_i = 1$)

Although the parameter s increases with constant spacing, the distances between frames are unequal In practice, the magnitude of the tangent vector is a second-order polynomial of the parameter s with its coefficients determined by the positions of four nearby key frames

It is important to note that the shape of the curve itself does not define the tangential magnitude at a certain point on the curve. Although different interpolation schemes, and thereby a different mathematical representation may produce the same curve shape, the tangential magnitude at a point of the curve varies depending on the interpolation scheme In other words, the magnitude of a tangent is not determined from the shape of a curve It is not an inherent property of the curve. In this respect, the tangential property of the parametric representation of a curve can be distinguished from the non-parametric curve For instance, the tangent of two-dimensional non-parametric curve can be represented as dy/dx with x, y being Cartesian coordinates. The magnitude of the tangent can readily be visualized such that it is large with steep slope and vice versa. This is not true for parametric curves, since the tangential magnitude is a function of the underlying implicit parameter s. Hence the visual slope, dy/dx in a parametric curve does not necessarily coincide with the magnitude of a tangent.

However, it is possible to relate this non-uniform tangent magnitude to the inter frame distance to achieve control of the motion speed. Since our approach is based on the discretization of the speed profile (Equation 4 1), the parameter (Equation 4 2), and the chord length (Equation 4.3), the continuous relationship in Equation 4 4 can be easily converted into a discrete form.

$$\Delta L_i \;=\; |\,\mathbf{d}_i\,|\;\Delta s_i \tag{4.6}$$

$$L = \lim_{n \to \infty} \left(\sum_{i=0}^{n-1} \Delta L_i \right)$$

$$= \sum_{i=0}^{n-1} \Delta L_i \qquad (4.7)$$

where

L is the total arc length between key frames,

$\mathbf{d}_i$ is the tangent at the ith inbetween frame whose position is determined by the interpolatory function $\mathbf{P}(s)$ at $(s = s_i)$

ΔL_i is the distance from ith to (i+1)th inbetween frames

The direct consequence of Equation 4.6 is that the chord length between frames is directly proportional to the magnitude of the tangent and the incremental parameter spacing. For instance, in Figure 4-7, we saw that the chord length (i.e., inter-frame distance) was proportional to the magnitude of the tangent with constant incremental parameter spacing. This means that, to maintain a constant speed, the parameter spacing should be reduced if the tangent value is relatively large.

Through use of Equation 4.7, the total arc length between key frames can be approximated by the sum of the piecewise line segments ΔL_i and this summation approaches the total arc length as the number of inbetween frames increases. Figure 4-8 illustrates an approximation of the total arc length between key frame A and B by three piecewise line segments. At the inbetween frame #3 (I.F. #3), the incremental arc length ΔL_3 can be roughly approximated by $|\mathbf{d}_3| \Delta s_3$. The direction of the tangent $\mathbf{d}_3$ approaches more closely the vector (I.F. #3-I.F. #2), as the number of inbetween frames increases. As a matter of fact, the proximity between the chord length and arc length is not important, since what we perceive as the speed in an animation sequence is the chord length between frames and not the arc length What matters, however, is how to control the incremental chord length to achieve proper speed.

In order for the incremental chord length, ΔL_i to incorporate speed information, three constraints should be applied as follows:

1 It should be proportional to the total arc length.

2. It should be proportional to the discretized speed value assigned to that frame.

3. The summation of the individual incremental arc length should be close to the total arc length as the number of inbetween frames increases

To satisfy these constraints, the expression for the incremental chord length is

$$\Delta L_i = L \frac{\phi_i}{\sum_{i=1}^{n} \phi_i} \qquad (4.8)$$

where

L is the total arc length,

ϕ_i is the discretized speed value at the i^{th} frame,

n is the number of inbetween frames including the beginning key frame

That is, the discretized speed value is normalized by the sum of individual speed values, so that Equation 4 7 is valid upon summing both sides of Equation 4 8 over n frames

Combining Equation 4.8 with Equation 4.6, we get the incremental knot spacing at the i^{th} frame as,

$$\Delta s_i = L \frac{1}{|d_i|} \frac{\phi_i}{\sum_{i=1}^{n} \phi_i} \tag{4.9}$$

with the tangent

$$\mathbf{d}_i = \left. \frac{dP(s)}{ds} \right|_{s = s_i}$$

where

L is the total arc length between key frames,

P(s) is an interpolation function

The incremental knot spacing at the ith frame is set to be inversely proportional to the tangents of the curve at that frame. The magnitude and direction of the tangents vary depending upon the interpolation scheme and the position of the key frames to be interpolated. However, in view of the fact that the great tangential magnitude amplifies inter-frame distance and thus the speed, the inverse relationship between the knot spacing and the tangent magnitude assures constant speed. This happens because, if the incremental knot spacing is diminished, the inter-frame distance tends to decrease, as can be verified by Equation 4.6. In contrast, the incremental knot spacing is set to be proportional to the discretized speed value at that frame. This setting is quite natural since the desired speed should be applied proportionally to the inter-frame distance.

Consequently, control of speed can be accomplished by adjusting the incremental knot spacing Δs according to the desired speed profile curve. In the beginning, the knot value s is set to zero to produce the first inbetween frame which in practice is a key frame. Then Equation 4.9 dictates the incremental knot spacing Δs to be applied to the current s value. The summation of (s+Δs) is the next knot parameter value corresponding to the second inbetween frame, and so on. Therefore, for any two adjacent frames, the distance from the first frame to the second is determined from the knot value of the first frame using incremental knot spacing.

In summary, this section has described the incremental knot spacing method for speed control in animation. Because of the difficulties involved in evaluating the displacement between frames, an approach to control speed by exploiting the tangent magnitude is presented. Based on the continuous representation of arc length, we have derived a discrete-domain counterpart as an approximation tool. Furthermore, the speed profile curve has been incorporated into the knot spacing expression.

Lookahead Adjustment

One of the problems associated with Equation 4.6 and Equation 4.9 is that it is valid only for sufficiently small values of the incremental knot spacing, Δs. For large values of Δs, the distance between two adjacent inbetween frames will increase and the tangent may not properly approximate the curve behavior between the frames. This is true especially when the number of inbetween frames decreases so that the total arc length is approximated by only a few line segments. The problem is undersampling in which the each sample represents the position of inbetween frames.

Figure 4-9 illustrates an undersampling error when three inbetween frames are used to interpolate a motion path F(s). At inbetween frame #2 (I.F. #2), Equation 4.8 was used to evaluate the distance from that frame to the next frame (I.F. #3) to control the motion speed. Then from Equation 4.5, the instantaneous tangential magnitude was calculated by differentiating the curve F(s). These two constraints determine the incremental knot spacing Δs so thet the knot value s_3 corresponding to (I F #3) can be estimated. However, upon insertion of the knot value into curve **P**(s), the point on the curve corresponding to the knot value may not be in the desired position (i e , I.F #3) The point may be before or after the (I.F #3) and there is no way to predict the position of the **P**(s_3) on the curve.

This error is a result of the assumption thet the tangential magnitude is constant between frames In the above example, the magnitude and direction of the $\mathbf{d}_2$ were assumed to be constant throughout the interval [I.F #2, I.F #3] and our method predicts the position of the inbetween frame #3 at (I.F #3) which no longer resides on the curve **P**(s) If a sufficient number of inbetween frames are used, and thus if the interval distance is small, the tangents can properly represent every instantaneous tangent attainable from the curve points in that interval. Since the general cubic spline is based on polynomials which are infinitely differentiable at all points on the curve, and since the tangential magnitudes can be represented by the square root of the second order polynomial of knot parameter s, it is reasonable to assume that the tangential magnitude is constant within a sufficiently small value of Δs. However, this is not true if the inbetween frames are sparsely spaced The longer the interval between frames, the greater the chance that the curve exhibits a change in magnitude.

Figure 4-10 shows a schematic diagram of Lookahead adjustment. The knot value corresponding to the inbetween frame #3 as evaluated in Figure 4-9 is inserted into the interpolatory curve **P**(s) so that the positional vector **P**' a candidate of the inbetween frame #3, is generated.

$$\mathbf{P}' = \mathbf{P}(s)\,\big|_{s = s_3}$$

Note that the trial position P ran past the desired position P_3. This situation may arise when the tangential magnitude at position P_2 is underestimated That is, the actual tangent value has increased in the interval $[P_2, P_3]$, and the tangent at P_2 has failed to represent all the tangents in that interval Due to an underestimation of the tangential magnitude, the incremental knot spacing Δs_2 was overestimated and thus the trial position ran past the position of the third inbetween frame, P_3.

Instead of using the trial position P_3 as the position of the frame, Lookahead adjustment uses the trial position as a basis to generate the proper frame position. In Figure 4-8, if we denote the distance from position P_2 to trial position P as ΔL_2, the general expression for the relation between them is

$$\Delta L_i = | P_{i+1} - P_i |$$

where

ΔL_i is the distance from the i^{th} frame to the $(i+1)^{th}$ trial frame,

P_{i+1} is the position of the $(i+1)^{th}$ trial frame,

P_i is the position of the i^{th} frame

One way to exploit this information about the distance from the current position to the next trial position is by scaling Let the knot spacing originally used to estimate the trial position be Δs_i and the new knot spacing to be evaluated be Δs_i Then we constrain tha new knot spacing such that Equation 4 6 is still valid

$$\begin{aligned} \Delta s_i \ \Delta L_i &= d_i \\ &= \Delta s_i \ \Delta L_i \end{aligned}$$

where

d_i is the tangential magnitude at the i^{th} frame.

With respect to Δs,

$$\Delta s_i = \Delta s_i \frac{\Delta L_1}{\Delta L_i} \qquad (4\,10)$$

If we regard the ratio of the original chord length to the trial chord length as a scale factor, then the new knot spacing is the trial knot spacing value multiplied by the scale factor If the original chord length specified by the speed profile curve is the same as the trial chord length, then the scale factor is unity and the original knot spacing value is maintained If, on the other hand, the trial chord length is greater than the original chord length, the new knot spacing value is scaled down, reflecting the difference in chord length.

Equation 4.10 can be combined with Equation 4.9 for an evaluation of knot parameter value that could control the speed of animation more accurately From the position of a current frame, the next frame position can be evaluated in the following way

1 Equation 4 9 is used to get the incremental knot spacing at the current frame.
2. Incremental knot spacing is added to current knot value to produce a trial knot value
3 The trial knot value is inserted into the expression for the interpolatory curve to yield the trial chord length
4 The trial chord length is incorporated into Equation 4.10, giving rise to a new incremental knot spacing
5 The new incremental knot spacing will be added to the current knot value and the result will be inserted into the interpolatory scheme to produce the position of the next frame

As the number of inbetween frames increases, the errors at the individual intervals between frames can also be diminished by this approach. Although the errors due to constant tangent representation at a certain interval can be greatly reduced with this approach, there still remains the problem of errors between the position calculated from our approach and the position dictated by the profile curve. While this section has dealt with reducing the positional errors between two adjacent frames, the next section will treat the cumulative errors associated with the total inbetween frames between a pair of key frames.

Averaging Adjustment

Using Equation 4.9, the spatial distance between adjacent frames and thus the speed between frames can be calculated based on the speed profile curve. In particular, if the motion path between a pair of key frames is inbetweened by a limited number of frames, application of a uniform tangent value between adjacent frames leads to the error in the spatial distance. This error is defined to be the distance gap between the inter-frame distance calculated by the incremental knot spacing and the one specified by a speed profile curve. Although Lookahead Adjustment could reduce the error between any two adjacent frames, the remaining errors will accumulate throughout all the inbetweened frames. A direct consequence of this cumulative error is that the key frame might be missed and it may not be displayed as part of an animation sequence. Since our approach is based on the principle that the key frame must be displayed during animation, the problem must be handled properly. This section resolves this problem by redistributing the cumulative errors into individual frame distances.

Before delving into further discussion, it is important to re-examine the relation between the frames and the normalized knot parameter s. Given a pair of key frames to be inbetweened, the position of the first key frame is assigned to be mapped from the parameter value of zero while that of the second key frame is assigned to be mapped from the parameter value of one. The assignments are made possible by applying these constraints into the expression for the splines representing the motion path. For instance, if three adjacent key frames K.F. #1, #2, #3 together with a pair of splines $P_1(s)$, $P_2(s)$ are given, the knot parameter s will run from zero to one to fill in the region [K.F. #1, K.F. #2] based on the function $P_1(s)$. Once again, the value of the knot parameter s, which varies between zero and one, will be inserted into $P_2(s)$ to produce the inbetweened frame positions. Although K.F. #2 belongs to both splines, it is normally treated as part of the second spline. Therefore, inbetweening a pair of key frames starts with s value of zero, which is the first key frame, and ends with designating the last inbetweened frame position right before the second key frame.

Since Incremental Knot Spacing is based on the discrete knot spacing Δs, and since the knot parameter s is assumed to be normalized, the sum of Δs should be exactly one. Only if the sum is identical to one, can the second key frame be properly positioned in reference to the last inbetweened frame, although the actual generation of the key frame is performed through the evaluation of the spline in the subsequent interval. This is true since the second key frame is assigned to be mapped from the knot parameter value of one in the normalized parameter representation of a cubic spline. The reason why Averaging Adjustment is needed is because the sum of the discrete knot parameter spacing Δs tends to deviate from one as the number of inbetween frames is reduced.

Figure 4-11 illustrates the Averaging Adjustment process. The positional vectors of the three inbetween frames (I.F. #1 theough I.F. #3), including the first key frame A (I.F. #1), are represented by $\mathbf{P}_i$ with $(i = 1, 2, 3)$. The frame positions calculated by Incremental Knot Spacing followed by Lookahead Adjustment are shown as $\mathbf{P}_i$, while the frame positions resulting from Averaging Adjustment are denoted by the positional vectors $\mathbf{P}_i$. Consider the position $\mathbf{P}_3$ of the inbetween frame I.F. #3 which is the last inbetween frame generated by the spline $\mathbf{P}(s)$. The key frame B is treated as the first inbetween frame of the subsequent interval generated with the knot parameter s value of zero. Therefore, Incremental Knot Spacing method will cease to evaluate the next frame position at point $\mathbf{P}_3$. If, however, the method continues to evaluate the next frame position $\mathbf{P}_4$, the calculated position must be the key frame B. Since Incremental Knot Spacing method in this case has divided the range of the normalized parameter s (i.e., $0 \leq s < 1$) into three subintervals Δs_1, Δs_2, and Δs_3, the knot parameter s_4 as obtained by adding Δs_3 to s_3 should be one. In light of the fact that key frame B is assumed to be mapped from s value of one, the knot parameter s_4 should become one to fulfill the assumption.

Unfortunately, the sum of the incrementel knot values and thus the value of the knot parameter s_4 generally does not become one. Due to the accumulation of errors between adjacent frames, the should-be-one knot parameter s_4 may exceed or fall below one. The errors in the evaluation of the distance between frames come from the errors in assigning proper knot parameter s values. The errors in parameter space are propagated and accumulated, since the value of s_2 is calculated from the Δs_1 and the value of s_3 is calculated from s_2 and Δs_2 and so on. The cumulative errors in the parameter space are represented by the parameter value of key frame B

Figure 4-11 shows the case when the parameter s_4 is much greater than one. If one maps this value into spatial position via a spline P(s), the corresponding position P_4 is far away from the desired position of the key frame B. In a strict sense, the spline P(s) with normalized parameter s, is undefined outside the interval [0, 1], but a value still exists since it is a polynomial. However, we do not need to evaluate this position in our approach.

Averaging Adjustment redistributes this cumulative error into individual inter-frame distances between adjacent frames. That is, if the should-be-one parameter value s_4 exceeds one, the preceding parameter s_1, s_2, s_3 will be reduced. Similarly the parameters will be scaled up if the parameter corresponding to the key frame B falls short of one. In doing so, we scale the incremental knot values in direct proportion with their magnitudes so the sum of adjusted incremental knot values becomes one

$$\Delta s_i = \frac{\Delta s_i}{\sum_{i=0}^{n-1} \Delta s_i} \tag{4.11}$$

where

n is the number of inbetween frames

Geometrically, scaling the knot parameters this way will shift the position of the key frames to the left or right of the previous frame positions. Figure 4-11 is an example of when the frame positions are shifted left, reflecting the reduction of the incremental knot value. Note that the correct position of key frame B now can be produced with s value of one, although the evaluation of that key frame is still left to the inbetweening of the subsequent interval.

It is by this adjustment of the incremental knot values such that they sum up to one, that the key frames are properly incorporated as part of an animation sequence For example, in a sequence of three key frames, each key frame can be seen by the insertion of a value of zero to the corresponding spline expressions Nevertheless, the position of the last inbetween frame is apt to be abnormal without Averaging Adjustment In Figure 4-11, the last inbetween frame position P_3 expects key frame B to be in the position P_4, according to a given speed profile at that point. Even if one can incorporate key frame B into the animation sequence by inserting zero s into the next interval, the resulting distance between the last inbetween frame and key frame B is far from the expected speed as examined before By relieving and averaging out the cumulative error at the position P_3, the transition from the last inbetween frame and the next key frame is smoothed out

Because of the fairing of the transition, our approach can span across multiple key frames While the inbetweening in [Bar89] treated entire keyframe sequences as a single pair of key frames, starting with the first key frame and ending with the last key frame, our approach splits the entire keyframe sequence into adjacent pairs of key frames and provides proper transition between keyframe intervals The main advantage of such splitting is that each pair of key frames can be filled with inbetween frames with independent speed control without affecting the speed of nearby key frames Each interval between a pair of key frames with discretized speed values is inbetweened while varying the normalized knot parameter between zero and one, and the burden of inbetweening entire key frames with missed key frames can be avoided. These points will be explained further in conjunction with the subsequent section about performance evaluation.

Although Averaging Adjustment does enable smooth transition to the next key frame, there is a price to be paid for such convenience We cannot foretell the sum of the incremental knot values a priori The denominator of

Equation 4.11 is not known until the Incremental Knot Spacing method completes evaluation of the entire series of inbetween positions Therefore, it is necessary to let the Incremental Knot Spacing run in the first pass of the program and the Averaging Adjustment take place in the second pass The next section will show how to implement this procedure in detail

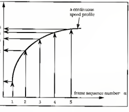

Figure 4-3. A speed profile curve $\phi(\alpha)$ represents speed values (i.e., inter-frame distance), the frame number α can take discrete values only

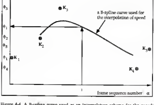

Figure 4-4 A B-spline curve used as an interpolation scheme for the speed. The speed ϕ_1 through ϕ_5 represents the speed values at corresponding key frames K_1 through K_5. Note that K_i does not denote spatial position. In addition, the interpolated speed values at key frames do not coincide with the specified value because of the approximating character of the B-spline.

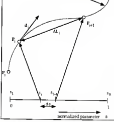

Figure 4-5. Relation between Δs_i, ΔL_i, and $\mathbf{d}_i$. $\mathbf{P}_1$ and $\mathbf{P}_n$ are positions of a pair of key frames in three-dimensional space, $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ are the positions of two consecutive intermediate frames. The positions were determined by a spatial cubic spline interpolation with corresponding knot parameters in s domain. The vector $\mathbf{d}_i$ is the tangential direction at i^{th} frame positioned at $\mathbf{P}_i$ on the curve P(s)

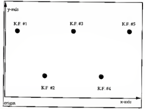

Figure 4-6 A template location of the key frames K.F. #i denotes the position of the key frame #i in two-dimensional space. For illustrative purpose the regions between K.F #2 and #3, and K.F #3 and #4 will be shown in subsequent examples

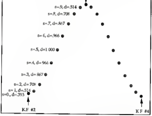

Figure 4-7 Variation in the magnitude of the tangent vectors $|\mathbf{d}_i|$. The black dots represent key frames while the gray dots represent the positions of the interpolated inbetween frames. The normalized parameter s runs from zero to one with constant increments (i.e., 0.1), while the frames run from the key frame #2 to #3. The magnitudes of tangent vectors are normalized such that the frame at (s = .5) has the value of 1.

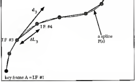

Figure 4-8. Approximation of the total arc length by the sum of incremental chord length ΔL_i. The spline P(s) interpolates key frames A and B. Key frame A is numbered as inbetween frame #1 (I.F #1). The total arc length from key frame A to B can be approximated by the sum of ΔL_i, $(i = 1 \ldots 7)$

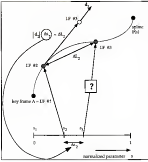

Figure 4-9 Approximation error due to undersampling Knot values ($s_1 = 0$) and ($s = 1$) map into key frames A and B, respectively Three inbetween frames including key frame A were generated to interpolate key frames A and B The inbetween frame I.F #2 was mapped from the knot parameter s_2 The distance from I.F #2 to I.F #3 was determined to be ΔL_2 by a speed profile curve The tangential magnitude at I.F #2, $\mathbf{d}_2$ is calculated from the differentiation of the curve P(s) with respect to s Based on these constraints, Δs_2 was computed using Equation 4 9 and added to s_2 to produce s_3 However, the frame position mapped from s_3 is not necessarily located at I.F #3 lying on the curve

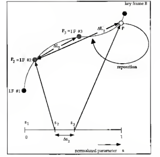

Figure 4-10 The Lookahead Adjustment. Knot values ($s_1 = 0$) and ($s = 1$) map into key frames A and B, respectively. The positional vectors $\mathbf{P}_2$, $\mathbf{P}_3$ represent the position of inbetween frames #2 and #3. The positional vector $\mathbf{P}$, the point on the curve P(s) produced by the incremental knot spacing Δs in Figure 4-9, could be further adjusted to be located near point $\mathbf{P}_3$ by a Lookahead Adjustment.

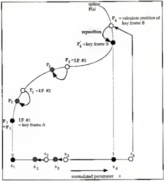

Figure 4-11 The Averaging Adjustment Original frame positions P_i and the frame positions resulting from Averaging Adjustments P_i are produced from the corresponding knot parameter values s_i and s_i. The spline $P(s)$ interpolates the motion path between key frames A and B Key frame A is treated as the first inbetween frame as mapped from a knot parameter value of zero.

Pseudo Code Representation

Incremental Knot Spacing, Lookahead Adjustment, and Averaging Adjustment together make up our approach to controlling the motion speed in animation Although each of them has been expressed explicitly using equations, we need to represent the methods from the point of view of programming to further clarify the approach.

It can be said that Lookahead Adjustment fine tunes the incremental knot values locally while Averaging Adjustment rearranges the incremental knot values in a global way At a certain position of an inbetween frame, Lookahead Adjustment prohibits the next frame position from exhibiting too much deviation from the distance specified by the speed profile curve Therefore, it applies to the distance between two adjacent frames On the other hand, Averaging Adjustment applies to the positions of the entire series of inbetweened frames.

Although Averaging Adjustment must be run in the second pass of a program, the previous knot values and the the incremental knot values can be directly restored from the first pass of the program Therefore, most of the procedures associated with the first pass, such as the evaluation of speed, chord length and tangents, can be avoided to speed up the second pass. The following Pascal-like pseudo code is made up of four parts: input specification, output specification, the first pass, and the second pass. Incremental Knot Spacing and Lookahead Adjustment are incorporated into the first pass. In particular, the Lookahead Adjustment shows a possible iteration to further reduce speed errors Lookahead Adjustment is incorporated into the second pass of the program titled as "Knots Controlled By The Speed Profile."

Program KnotsControlledByTheSpeedProfile.

```
BEGIN
Input:
    PointList = spatial position of four sample data points
    TypeSpline = spline type to determine shape
    TypeSpeed = a method to assign a speed profile
                It can be a function or a SpeedList with a spline type to
                interpolate the speed
    NumFrames := number of inbetween frames
    NumIterations = number of iterations for Lookahead Adjustment

Output:
    The spatial position of the inbetweened frames including the
    beginning key frames: Spline[ ]

First pass:
    FOR i=1 to (NumFrames-1)
    DO BEGIN
        KnotList[i] = i/NumFrames
        /* initialize the s parameter with equal knot spacing */
    END
    TotalChordLength = SumChord(KnotList,TypeSpline)
    /* sum chord length with constant Δs, */
```

```
TotalSpeedQuantity = SumSpeed (SpeedList,TypeSpeed,KnotList)
/* sum $\varphi_i$ mapped from knot values spanned from constant $\Delta s_i$ */
SumSi = 0
/* the variable to be passed into the second pass is initialized to zero */
FOR i=1 TO NumFrames
DO BEGIN
/* Incremental Knot Spacing method follows*/
        Si := KnotList[i]
        CurrentLocation := GetLocation(TypeSpline,PointList,Si)
        /* Find the frame position matching $S_i$ */
        Derivative = GetDerivative(TypeSpline,PointList,Si)
        /* Find the tangential magnitude $| d_i |$ at current position */
        Speed = GetSpeed(TypeSpeed,SpeedList)
        /* Evaluate the discretized speed value by a function or direct
        assignment */
        DeltaSi = (TotalChordLength/Derivative) *
                    (Speed/TotalSpeedQuantity)
        /* incremental knot spacing is determined */
        FOR j=0 TO NumIteration
        /* the comparison between the trial chord length and the
           specified speed profile can be iterated */
        DO BEGIN
                TriedSi := Si+DeltaSi
                /* Get the knot value of the next frame */
                TriedLocation = GetLocation(
                                TypeSpline,PointList,TriedSi)
                /* Find Lookahead position mapped from $s_i+\Delta s_i$*/
                LocationDifference := AbsoluteValue
                                (TriedLocation-CurrentLocation)
                /* Evaluate the error in the speed */
                DeltaSi := DeltaSi *
                        (DeltaSi*Speed)/(LocationDifference)
                /* Lookahead adjustment is complete */

        END
        SiList[i] = Si
        /* Save for second pass */
        DeltaSiList[i] = DeltaSi
        /* Save for second pass */
        SumSi = SumSi+Si
        /* Save for second pass */
        KnotList[i+1] := Si+DeltaSi
        /* Adjust next knot location */
    END
```

Second Pass:

```
    SumSi = SumSi
    /* Recovered $\sum_{i=0}^{n-1} \Delta s_i$ from first pass */
    Spline[0] = GetLocation(TypeSpline,PointList,S0)
    /* first point of the frames */
    FOR i=2 TO NumFrame
    DO BEGIN
        Si := SiList[i]
        /* Recovered $s_i$ from first pass */
        DeltaSi = DeltaSiList[i]
        /* Recovered $\Delta s_i$ from first pass */
        DeltaSi = DeltaSi/SumSi
        /* Averaging adjustment */
        Si = Si+DeltaSi
        /* Get new Si */
        Spline[i] := GetLocation(TypeSpline,PointList,Si)
          /* Locate the frame positions with the new knot values */
    END
END
```

Error Analysis of the Algorithms

This section illustrates visually how Incremental Knot Spacing and the subsequent adjustments enhance the control of Inter-frame distances The cootrol of the inter-frame distance can equally be said to be control of speed, since speed in our approach is defined as the distance between frames To facilitate the understanding of positional variations of the frames under our approach, the adjustments are tested against the implementation of the constant speed between adjacent frames

All the figures shown in this section are based on the two-dimensional template data positions (i e., keyframe positions) in Figure 4-6. As a result, the frame positions appearing in the subsequent figures denote the positions in the two-dimensional Cartesian coordinate system. Two-dimensional data is used only for the purpose of convenience for illustration, and the adjustment techniques are equally applicable in the three-dimensional spatial data. The positions of the produced inbetween frames are shown only in the region between the three inner data points of Figure 4-6. Throughout the figures in this section, the gray dots represent the position of inbetween frames and the black dots represent those inbetween frames that also belong to key frames. The positional vector $\mathbf{P}_i$ represents the location of the frames.

Before showing the speed adjustment techniques, a comparison is made between the performance of the tensioned cardinal spline and the free form spline discussed in Chapter 3. The purpose of the comparison lies in the fact that the speed and thus the inter-frame distance is determined from the representation of the splines interpolating the motion path rather than the shape of the spline curves. The same curve can yield a different pattern of the inbetween frames depending on the mathematical representation of the curve.

Figure 4-12 and Figure 4-13 illustrate the location of the inbetween frames produced by constant knot spacing. Ten inbetween frames, including the first key frame at $\mathbf{P}_1$, are generated to fill in the motion path from the key frame to the next key frame at $\mathbf{P}_{11}$. Similarly, the inbetween frames between the key frames at $\mathbf{P}_{11}$ and $\mathbf{P}_{21}$ were generated while varying the knot parameter s from zero to 9 with constant incremental knot spacing Δs. Both figures illustrate a highly-tensioned motion path such that the paths are linear between adjacent key frames. Although the same constant knot spacing was used to generate the inbetween frames in the two identical motion paths, the spatial pattern of the distribution of the inbetween frames is quite different. The tensioned cardinal spline [Sm83] in Figure 4-12 shows a relatively dense frame distribution near the key frames at $\mathbf{P}_1$, $\mathbf{P}_{11}$ and $\mathbf{P}_{21}$, while it shows sparse distribution near the center of the adjacent key frames As a result, the motion appears to be decelerating near the key frames, when compared with the motion near the center In contrast, the free form spline in Figure 4-13 exhibits almost uniform distribution near the key frames The constant knot spacing has led directly to a constant spacing between the frames, thereby producing constant motion speed

The different behaviors can be verified using the relation between the knot parameter s and the spatial position of the frames P(s) mathematically The tensioned cubic cardinal spline is a modification of conventional cardinal spline and is expressed as,

$$P(s) = [s^3\ s^2\ s\ 1]\ T\ [P_1\ P_2\ P_3\ P_4]^T$$

with the tension matrix

$$T = \begin{bmatrix} -a & 2-a & a-2 & a \\ 2a & a-3 & 3-2a & -a \\ -a & 0 & a & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

where

a is a tension control parameter,

P(s) is the curve interpolating the region $[P_2, P_3]$

while the knot parameter s varies between zero and one, so that $P(s)\big|_{s = s_t}$ is the inbetween frame position corresponding to $(s = s_t)$

The tangent vector of this spline is,

$$\frac{dP(s)}{ds} = [3s^2\ 2s\ 1\ 0]\ T\ [P_1\ P_2\ P_3\ P_4]^T$$

Let us focus on the first two matrices in the right hand side of the above equation, the matrix [3s2 2s 1 0] and T Even if the tension parameter a is set to one for high tension, the elements in the first row of tension matrix **T** do not vanish. Therefore, the term s^2 is still involved in the expression for the derivative of the curve **P**(s) upon multiplication of the first two matrices. That is, the magnitude of the tangent on some point on the curve **P**(s) becomes a second order function of s. In terms of the distance between frames, the inter-frame distance per unit s is a second order polynomial of s.

In contrast, taking the tangent of the free form spline (which does not necessarily belong to the G^1 class), one obtains

$$\frac{dP(s)}{ds} = [3s^2\ 2s\ 1\ 0]\ \mathbf{A}\ [\mathbf{P}_1\ \mathbf{P}_2\ \mathbf{P}_3\ \mathbf{P}_4]^T$$

with

$$\mathbf{A} = \begin{bmatrix} -u+2-\frac{1}{u} & -1+\frac{1}{u}+v & u+\frac{1}{v-1} & -v-1+\frac{1}{1-v} \\ 2u-4+\frac{2}{u} & 2-\frac{2}{u}-v & -2u+1+\frac{1}{1-v} & v+1-\frac{1}{1-v} \\ -u+2-\frac{1}{u} & -2+\frac{1}{u} & u & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

where

u, v are the midpoint knot parameters corresponding to point $\mathbf{P}_2$ and $\mathbf{P}_3$ respectively

Each of the elements of first two rows of matrix **A** becomes close to zero if u equals one and v approaches zero. The third row of matrix **A** combines with the P matrix to produce the constant tangent vector $(\mathbf{P}_3\text{-}\mathbf{P}_2)$. Therefore, the tangent vector is,

$$\frac{dP(s)}{ds} = \mathbf{P}_3\ \mathbf{P}_2$$

Since the tangent is defined as the displacement in $\mathbf{P}(s)$ with unit s, the displacement between the positions $\mathbf{P}(0)$ and $\mathbf{P}(1)$ is the same ss the magnitude of the vector $(\mathbf{P}_3\text{-}\mathbf{P}_2)$ if $\mathbf{P}(s)$ is linear. Since the magnitude of the tangent is constant regardless of the knot value s, and since the curve is strictly linear, the inter-frame distance is alwaya uniform with the highly-tensioned free form splines. Therefore, it can be said that the spatial distribution pattern of the frames on s interpolatory curve depends on msthematical representation rather than the shape of the interpolatory curve.

However, if the u, v parameters change from the values causing the high tension, the elements in first two rows of the matrix **A** will be certain values other than zero. Consequently the non-zero rows will cause the s^2 term and the s term to be part of the expression for the tangents. The same holds true for the tensioned cardinal spline. Then constant knot spacing in the knot parameter domain no longer assures constant spacing in the space domain. It msy produce unwanted acceleration and deceleration due to the non-uniform distribution of the frames on the motion path, thus necessitating the use of the speed control techniques developed earlier.

Figures 4-12 through 4-15 show a gradual enhancement in controlling the motion speed when a constant inter-frame distance is desired. An ordinary cardinsl spline was used to generste the motion path of the figures. Figure 4-14 shows the inbetweened frame positions when no speed control strategy is taken. The knot parameter s domain is simply subdivided into equal knot spacing so thet the inbetween frame #1 (i.e., key frame), #2, #3, #4 and #5 are mapped from ($s = 0$, $s = .2$, $s = .4$, $s = .6$, $s = .8$) with corresponding positional vectors $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$, $\mathbf{P}_4$ and $\mathbf{P}_5$. Notice that the inter-frame distance varies in a non-uniform manner as a function of the parameter value s since the tangential magnitude is a second-order polynomial of the parameter s.

The positional vectors $\mathbf{P}_6$ through $\mathbf{P}_{10}$ are generated similarly and are shown here against the positional symmetry with respect to the frames $\mathbf{P}_1$ through $\mathbf{P}_5$ They will not be covered in the tables following the figures since the properties in the interval $[\mathbf{P}_6,\mathbf{P}_{10}]$ match exactly with those of the interval $[\mathbf{P}_1,\mathbf{P}_5]$

Table 4-1 shows relevant measures of the inbetweening in Figure 4-14 as follows.

1 The column labeled "Frame Number" indicates the sequential frame positions.

2 The column labeled "Tangential Magnitude" indicates the tangential magnitude $|\mathbf{d}_i|$ at a given frame position as defined in Equation 4 5 The numbers are represented relative to the distance from $\mathbf{P}_1$ to $\mathbf{P}_5$ which is set at 297

3 The column labeled "Inc Knot Value" indicates the incremental knot value, Δs_i, used to generate the next inbetween frame position.

4 The column labeled "Measured Distance" indicates the real inter-frame distance ΔL_i, as measured from the frame positions in the preceding figures using Equation 4 3

As can be seen in the "Tangential Magnitude" column, the speed per unit knot value ($s = 1$) is increasing from $\mathbf{P}_1$ toward the center of $\mathbf{P}_3$ and $\mathbf{P}_4$ and from then on decreases until the position $\mathbf{P}_5$ The "Measured Distance" column, representing the measured inter-frame distance in Figure 4-14, roughly reflects the trend in the tangential magnitude, since the incremental knot values are constant. The constant knot spacing in the parameter domain produces non-uniform spatial distance between frames with a mean of 61 and standard deviation of 18, where the distance is measured relative to the length between $\mathbf{P}_1$ and $\mathbf{P}_6$, which is set to 297 If corresponding inbetween frames are displayed sequentially, the speed picks up toward the middle of the interval $[P_3,P_6]$ and slows down toward the keyframe position P_6. Although the figure shows symmetrical speed centered near the middle of P_3 and P_4, there is no regular pattern to the acceleration and deceleration of the speed in general, as can be deduced from Equation 4.5. Depending upon the interpolation method and the location of the key frames, the tangential magnitude varies unpredictably and no general rule exists governing the behavior of the tangential magnitude that could facilitate control of the motion speed.

Figure 4-15 shows the inbetween frame positions upon application of Incremental Knot Spacing method. The total arc length L used in Equation 4.9 is assumed to be the same as in the ordinary cardinal spline, so that the mean value of the Measured Distance column of Table 4-1 is the inter-frame distance desired in this case. A more accurate value of the arc length can be estimated by subdividing the arc with more line segments. However, the arc length thus computed does not necessarily coincide with the sum of the inter-frame distances, especially when fewer inbetween frames are required, and it is reasonable to crudely estimate the total arc length as the sum of the inter-frame distances with a given number of inbetween frames using the constant incremental knot value. The same number of inbetween frames as in Figure 4-14 are generated, but the method diversifies the incremental knot values rather than using constant incremental values to produce uniform inter-frame spacing. The speed is controlled by manipulating the incremental knot value Δs_i and thus the knot value s_{i+1}, as obtained from $(s_i+\Delta s_i)$, corresponding to the $(i+1)^{\text{th}}$ frame. In Table 4-2, two new columns are defined as follows:

1. The column labeled "Calculated Distance" indicates the desired

inter frame distance ΔL_i, incorporating the speed profile using Equation 4.6 and 4.9.

The numbers are shown relative to the distance from P_1 to P_5, which is set to 297

2. The column labeled "Distance Error" indicates the error between the desired inter-frame distance and the resulting real inter-frame distance

To maintain a constant distance between frames, the elements of the column "Calculated Distance" are all set to a constant value 61. This setting is possible by adjusting the incremental knot values in association with the corresponding tangential magnitudes, since the calculated distance is the multiplication of the two columns. For instance, because of the small tangential magnitude at the first frame position P_1, incremental knot value is impelled to be relatively large at the position. Despite the adjustment, one can notice the relatively large displacement of P_2 from P_1 compared with other inter-frame distances, as tabulated in the Measured Distance column corresponding to position P_1. This happens because the interval $[P_1, P_2]$ is represented by a single constant tangential magnitude, although the actual tangential magnitude was increasing in that interval. As a result, the increased actual tangential magnitude was combined with the relatively large incremental knot value to yield a large distance gap. In particular, the change of the tangential magnitude (i.e., the second derivative of the interpolation curve P(s)) is very rapid near the frame position P_1 as can be seen in the Tangential Magnitude column of Table 4-2. Incremental Knot Spacing could not catch up with the rapid change in tangential magnitude with so few frames

The sum of the incremental knot values for the situation at hand exceeds one, the maximum value of the normalized knot parameter. This happens because the frame interval $[P_1,P_2]$ consumes too much of the total knot parameter space. Because of the large incremental knot value Δs_1, the subsequent knot values s_2 through s_5 tend to be pulled toward the key position P_6. In parallel with the knot values, the frame positions are also drawn toward the key position. One consequence of the overestimation in Δs_1 is that it propagates to the position P_5 where it tries to locate the next frame P_6 to a position further away from the real frame position P_6, since the knot value of the next frame exceeds one. To maintain the constant inter-frame distance between P_5 and P_6, the incremental knot valus at P_5 should be assigned a value that causes the sum of the incremental knot value to exceed one. In practice, the position of P_6 is not evaluated as part of the inbetween frame in the interval $[P_1,P_6]$. Since it is the initial key frame of the interval $[P_6,P_{11}]$, it is evaluated with knot parameter ($s = 0$) in that interval and the position cannot be changed. Consequently, the inbetweened position P_5 comes near to the position P_6, which belongs to the key frames and the motion speed gets slowed down relative to the previous motion.

Comparing intervals $[P_1,P_6]$ and $[P_6,P_{11}]$, the symmetry of the frame positions with respect to position P_6 is not maintained. For the same reason that P_2, the position of the frame #2, is overly displaced from frame #1, frame #7 shows a large displacement from frame #6. Incremental Knot Spacing is dependent upon the path it takes to produce the knot values. For instance, in the interval $[P_6,P_{11}]$, if the knot values are evaluated starting from the keyframe position P_{11} up to frame P_6, frame #10 will show a large displacement from frame #11 and the frames generated in both intervals will exhibit exact symmetry.

Figure 4-16 illustrates the frame positions appearing after Lookahead Adjustment is applied to Incremental Knot Spacing in Figure 4-14. Notice that the incremental knot value of the first frame has been reduced as a result of Lookahead Adjustment Since the position of the second frame with Incremental Knot Spacing was overly displaced from the first frame, Lookahead Adjustment did look ahead to the second frame position and readjusted the incremental knot values The reduction led to a decrease of the Calculated Distance and thus the reduction of the Measured Distance Note, however, this produced a slightly smaller distance between the first two frames than the expected mean distance 61 As a result of this underestimation, the sum of the incremental knot spacing falls below one, as evidenced by the long distance between frame #5 and #6 The underestimation can be corrected upon iterative application of Lookahead Adjustment as suggested in the section on Pseudocode Representation Overall, the sum of the absolute distance error is reduced and the standard deviation from the mean value is greatly reduced, the inter-frame distance is close to constant compared with Incremental Knot Spacing That is, the application of Lookahead Adjustment further aids in controlling the motion speed, which is constant in this case

Although Lookahead Adjustment further facilitates the control of the motion speed, there should be a way to let the sum of incremental knot values be one to ensure proper distance between the last inbetweened frame in the first interval and the first frame in the next interval as described before. Figure 4-17 and Table 4-4 show the result of Averaging Adjustment applied to Lookahead Adjustment in Figure 4-16 The unduly elongated inter-frame distance between P_5 and P_6 in Figure 4-16 has now been redistributed through entire the inter-frame distance in the interval $[P_1,P_6]$ The redistribution corresponds to pulling all of the frame positions in $[P_1,P_6]$ toward frame #6 at P_6 In parallel with the distance adjustment, the incremental knot values are redistributed while assuring thet they sum to one By redistributing the distance errors over the entire interval, the transition from the last inbetweened frame (i.e , frame #5 at P_5 in this case) to the first inbetweened frame (i e , frame #6 at P_6) becomes smoother than is the case with no Averaging Adjustment. The smoothness near the key frames (frames #1, #6, #11 at positions P_1, P_6, P_{11}, in this case) is important since in our approach, the motion speeds in each pair of key frames are separately controlled and the frames are concatenated to produce frame positions spanning multiple key frames

Figures 4-18 illustrates an example in which the speed is controlled to vary between frames The same motion path as in Figure 4-17 is used to produce a total of 16 frames in Figure 4-18 Three discrete speed values, 8, 1, 8 are assigned for the key frames at P_1, P_{16}, and P_{31} respectively One can specify 4, .5, 4 for the same speed values however, since only the relative values of the speed count The inter-frame distance at the frame at P_9 is supposed to be eight times shorter than the one near the frames at P_1 and P_{17} To produce a gradual speed change, the inbetween frames are set to take the speed values calculated by linear interpolation, such that the speed at the $i^{\underline{th}}$ inbetween frame in the interval $[P_1,P_9]$ is given by

$$\begin{aligned}\phi_i &= (\text{speed of the starting key frame}) \frac{(i-1)}{\text{number of frames}} + \\ &\quad (\text{speed of the ending key frame}) \left(1 - \frac{(i-1)}{\text{number of frames}}\right) \\ &= 1\,\frac{(i-1)}{8} + 8\left(1 - \frac{(i-1)}{8}\right)\end{aligned}$$

where

the frame number i is an integer ranging from 1 to 8.

It is important to note that the temporal aspect is completely separated from the spatial aspect in our approach. The speed profile can be independently generated from the motion path. For instance, the motion speed is controlled by a linear interpolation method while the motion path is produced by the cardinal spline Incremental Knot Spacing followed by Lookahead Adjustment and Averaging Adjustment were used to generate Figure 4-18, in which the motion shows an explicit slowdown near the key frame positioned at $\mathbf{P}_9$ while it speeds up near the key frames positioned at $\mathbf{P}_1$ and $\mathbf{P}_{17}$. A measure of error, the difference between the Calculated Distance and the Measured Distance, sums to 34 as can be seen in Table 4-5. Most of the errors are distance errors near frame #1 positioned at $\mathbf{P}_1$ and this can be verified by comparing the speed profile curve and the measured speed in Figure 4-19. In the figure, the speed profile curve becomes linear with respect to the frame sequence number because of the linear speed interpolation scheme given above. However, the measured speed shows some deviation from the speed profile curve near the first frame and the error comes from the fact that the tangential magnitude is underestimated near the first frame, causing large incremental knot values.

Figure 4-20 shows the relationship between the frame sequence number and the knot values in Figure 4-19. As was mentioned previously, in our control scheme, the motion speed is controlled by adjusting the knot parameter value s corresponding to the sequential frame number. Roughly speaking, the inter-frame distance is proportional to the magnitude of the incremental knot spacing. In view of the fact that the slope at a certain frame in Figure 4-20 can represent the incremental knot spacing, the slope near frame #8 is small compared with that of frame #1. Therefore, the deceleration of speed in Figure 4-18 is driven by the decrease in the incremental knot spacing in Figure 4-20.

Figures 4-21 and 4-22 illustrate the position of inbetween frames and the measured speed values when the number of inbetween frames is double thet of Figure 4-18 The same motion path and method for the motion speed control was used in both figures As can be seen, the inter-frame distance errors including the frames near the first frame tend to be smaller and thus the measured speed tends to approach the speed profile curve Another measure of error, the sum of absolute distance errors, is much reduced compared with Table 4-5 In fact, the error approaches zero as the number of inbetween frames increases

Figures 4-24 through 4-27 illustrate how rapidly the errors can be reduced in our motion control method by increasing frame numbers with different motion pathe shown in Figure 4-23 The results in the figures are based on a speed profile with a constant speed, and therefore constant inter-frame distance. To compare the behavior of our speed control scheme in the longer interval $[P_2,P_3]$ and the shorter interval $[P_3,P_4]$, Figure 4-23 is made to be unsymmetrical As a measure of error, the sum of the incremental knot values is used in Figure 4-24 and 4-25 If Averaging Adjustment is employed, the sum is identically zero However, the sum can be used to estimate how accurately Incremental Knot Spacing and Lookahead Adjustment can approximate the given constant speed profile As long as it approaches one, the individual knot spacing and corresponding inter-frame distance can be said to be estimated properly As another measure of error, the sum of the Distance Error is used in Figures 4-26 through 4-27 Part of the reason why it is used instead of the sum of the absolute Distance Error is that it can be a criterion of whether the total Calculated Distance exceeds the Measured

Distance by including the sign. This error should approach zero as the number of inbetween frames increases, so that the Calculated Distance can properly approximate the Measured Distance.

Depending upon the interpolation scheme of the motion path, the performance of our speed control method varies if a relatively small number of inbetween frames are incorporated. The variation occurs since each interpolation scheme has a different mathematical representation of the motion path and the difference leads to variation in the tangential magnitude exploited in our speed control method. Figures 4-24 and 4-25 illustrate the sum of incremental knot values in the intervals $[P_2,P_3]$ and $[P_3,P_4]$, respectively. The values are taken when Incremental Knot Spacing followed by Lookahead Adjustment are applied to three different interpolating schemes for the motion path. At a small number of inbetween frames, the sum of the incremental knot values of the Type II free form spline exceeds one, while that of the cardinal spline and the B-spline falls below one. The cardinal spline falls further away from zero than the B-spline. This means that the Type II free form spline and the cardinal spline have a greater curvature in this region, causing the tangential magnitude to vary rapidly enough for a fixed tangent to fail to properly represent the entire range of tangential changes happening between a pair of inbetweened frames. Note that the tangential magnitude cannot be determined directly from the appearance of the curve shape in Figure 4-23. It can be evaluated only by mathematical representation, as mentioned earlier. In the interval $[P_3,P_4]$, however, only the sum of the incremental knot values of the Type II free form spline exhibits convergence to one before the number of inbetween frame reaches five, as can be seen in Figure 4-25. In view of the curvature, this rapid convergence complies with the general behavior of the G^1 free form splines explained in Chapter 3 G^1 free form spline has relatively constant curvature in the narrower interval so that a fixed tangent can be effectively used to estimate the tangential magnitude in each subinterval made up of the motion path between a pair of key frames Overall, the sum of the incremental knot values shows repid convergence to one with relatively few inbetween frames.

The distance errors in Figures 4-26 and 4-27 are the relative magnitudes of the distance when the distance between the keyframe positions P_2 and P_3 is set to one. Incremental Knot Spacing, Lookahead Adjustment and Averaging Adjustment were used to generate the inbetween frames leading to the figures. As the number of inbetween frames increases, the distance errors approach zero so that the calculated distance makes little difference to the measured distance In Figure 4-26, the sum of distance errors of Type II free form spline is to the negative side, while others are positive with a low number of inbetween frames This means that the calculated distance is smaller than the measured distance

It is interesting to compare the behavior of the curves with respect to the sum of the incremental knot values with thet of the curves with respect to the sum of distance errors For instance, if the sum of the incremental knot spacing of Type II free form spline in Figure 4-24 is greater than one, it is less than zero in Figure 4-26 Other curve schemes exhibit similar behavior The same is true for the Figures 4-25 and 4-27 As the sum of the incremental knot spacing becomes greater than one, the sum of distance errors tends to be less than zero The principal reason for such behavior is that Averaging Adjustment counteracts the error in the sum of the incremental knot values. If the sum of the incremental errors is large, Averaging Adjustment makes the sum into one by reducing the calculated distances to the point where the sum of the calculated distance is less than the sum of the measured distance. This is true in the case of a small number of the inbetween frames. However, as the number of frames increases, the counteracting property of Averaging Adjustment is small enough to be ignored and the distance errors approach zero. In conclusion, Incremental Knot Spacing combined with Lookahead Adjustment and Averaging Adjustment proves fairly effective in terms of the associated errors as the number of inbetween frames increases.

Summary

This chapter has been devoted to the control of motion speed in animation. In connection with our animation system, we developed a method to control the motion speed of a single atom so that the displayed frame sequences do not exhibit unnecessary acceleration, or even random motion dynamics. Clues involved in the protein folding can be visualized with ease if the motion is controlled such that a smooth transition between frames is assured. In this chapter, we have presented a flexible speed control scheme for that purpose.

The basic building block of our method is Incremental Knot Spacing, which dynamically adjusts the parameter values of adjacent inbetween frames, depending upon various parameters. Lookahead Adjustment is employed to avoid relatively large errors in the incremental knot values. Averaging Adjustment was introduced to secure a controlled transition from the last inbetweened frame to the next key frame.

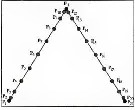

Figure 4-12. The positions of inbetween frames in a tensioned cardinal spline with a tension value of one. Incremental knot spacing Δs is set to a constant value of 0.1. The positional vectors P_1, P_{11} and P_{21} represent the key frame positions. Within region [P_1,P_{11}], P_1 through P_{10} are the positions of the inbetween frames produced by evaluating the cardinal spline at corresponding knot values (s = 0) through (s = .9)

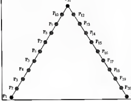

Figure 4-13 The positions of inbetween frames in a free form spline with midpoint knot parameter values of u = 1, and v = 0 (e.g. .025). Incremental knot spacing Δs is set to a constant value of 0.1. The positional vectors $\mathbf{P}_1$, $\mathbf{P}_{11}$ and $\mathbf{P}_{21}$ represent the key frames. Within region [$\mathbf{P}_1$,$\mathbf{P}_{11}$], $\mathbf{P}_1$ thorugh $\mathbf{P}_{10}$ are the positions of the inbetween frames produced by evaluating free form spline at corresponding knot values (s = 0) through (s = .9).

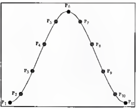

Figure 4-14 Inbetweened frame positions with constant knot spacing.

Frame Number	Tangential Magnitude	Inc Knot Value	Measured Distance
1	160	.200	41
2	288	.200	70
3	394	.200	81
4	394	.200	70
5	288	.200	41

Table 4-1 Inter-frame distances of Figure 4-14.
Sum of incremental knot value = 1 000,
Standard deviation of measured distance = 18 4

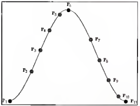

Figure 4-15 Inbetween frame positions upon application of Incremental Knot Spacing

Frame Number	Tangential Magnitude	Inc Knot Value	Calculated Distance	Measured Distance	Distance Error
1	160	.381	61	102	-41
2	388	157	61	63	-2
3	405	150	61	58	3
4	359	179	61	52	9
5	242	251	61	26	35

Table 4-2. Error analysis of Figure 4-15·
Sum of incremental knot value = 1 110, Sum of absolute distance error = 90, Standard deviation of measured distance = 27.4.

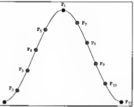

Figure 4-16. Inbetween frame positions upon application of Lookahead Adjustment

Frame Number	Tangential Magnitude	Inc. Knot Value	Calculated Distance	Measured Distance	Distance Error
1	160	.229	37	49	-12
2	309	.168	52	60	-8
3	309	151	59	61	-2
4	405	.159	64	61	3
5	350	.210	73	70	3

Table 4-3. Error analysis of Figure 4-16:
Sum of incremental knot value = 917,
Sum of absolute distance error = 28,
Standard deviation of measured distance = 7.5

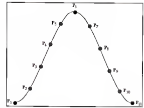

Figure 4-17 Inbetween frame positions upon application of Averaging Adjustment.

Frame Number	Tangential Magnitude	Inc Knot Value	Calculated Distance	Measured Distance	Distance Error
1	160	249	40	56	-16
2	323	183	59	68	-9
3	401	165	66	67	-1
4	395	173	68	62	-6
5	310	229	71	50	21

Table 4-4. Error analysis of Figure 4-17.
Sum of incremental knot value = 1 000,
Sum of absolute distance error = 53,
Standard deviation of measured distance = 7.6

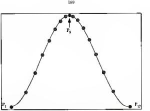

Figure 4-18 Linear speed interpolation with 8 inbetween frames in intervals $[P_1, P_9]$ and $[P_9, P_{17}]$

Frame Number	Discret. Speed	Tangential Magnitude	Inc. Knot Value	Calculated Distance	Measured Distance	Distance Error
1	8.0	160	.247	40	55	-15
2	7 12	322	164	53	60	-7
3	6 25	397	129	51	52	-1
4	5 38	405	111	45	44	1
5	4 50	376	100	37	35	2
6	3 62	323	091	30	26	4
7	2.75	255	084	21	19	2
8	1 88	190	073	14	12	2

Table 4-5 Error analysis of Figure 4-18 Sum of absolute distance error = 34

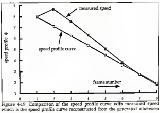

Figure 4-19 Comparison of the speed profile curve with measured speed which is the speed profile curve reconstructed from the generated inbetween frames in Figure 4-18

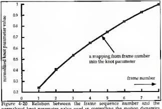

Figure 4-20 Relation between the frame sequence number and the normalized knot parameter value used in controlling the motion dynamics in Figure 4-19

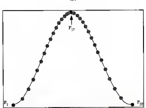

Figure 4-21. Linear speed interpolation with 16 inbetween frames in intervals $[P_1,P_9]$, and $[P_9,P_{17}]$.

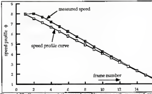

Figure 4-22. Comparison of the speed profile curve with measured speed which is the speed profile curve reconstructed from the generated inbetween frames in Figure 4-21

Frame Number	Discret Speed	Tangential Magnitude	Inc. Knot Value	Calculated Distance	Measured Distance	Distance Error
1	8 0	160	.159	25	31	-6
2	7.56	257	.104	27	31	-4
3	7 12	332	083	27	29	-2
4	6.69	375	071	27	28	-1
5	6 25	398	.064	25	26	-1
6	5 81	407	.059	24	24	0
7	5 38	406	055	22	22	0
8	4 94	395	052	21	20	1
9	4.50	378	.050	19	18	1
10	4.06	355	.048	17	16	1
11	3.62	328	047	15	15	0
12	3.19	296	045	13	13	0
13	2.75	261	.045	12	11	1
14	2.31	225	.043	10	9	1
15	1 88	192	.041	8	7	1
16	1 44	168	035	6	6	0

Table 4-6 Error analysis of Figure 4-21
Sum of absolute distance error = 20
Sum of individual lengths = 306.

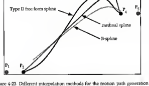

Figure 4-23. Different interpolation methods for the motion path generation. Position of key frames are represented by P_i. Figures 4-24 through 4-27 are based on the inbetween frames generated by these interpolation methods. The gray dots represent the position of the key frames used as end points for the interpolatory curves in the intervals $[P_2,P_3]$ and $[P_3,P_4]$.

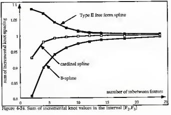

Figure 4-24. Sum of incremental knot values in the interval $[P_2,P_3]$

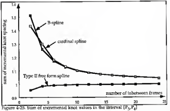

Figure 4-25. Sum of incremental knot values in the interval $[P_3,P_4]$

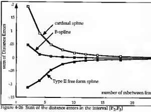

Figure 4-26 Sum of the distance errors in the interval $[P_2,P_3]$

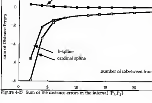

Figure 4-27 Sum of the distance errors in the interval $[P_3,P_4]$

CHAPTER 5
APPLICATION SCRIPTS FOR PROTEIN FOLDING

Introduction

This chapter describes an animation system as an application of the preceding chapters. More specifically, a prototype system is designed for the simulation of the protein folding. The free form splines developed in Chapter 3 form the basic building blocks of the animation system. First, it is used for producing a curve representing the backbone conformation, and second, it is used for the generation of a motion path of individual backbone atoms. The speed control techniques developed in Chapter 4 are used to generate a smooth motion transition between frames. In addition, inbetweening techniques discussed in Chapter 4 are introduced, for the first time in the field of molecular graphics, in the context of the backbone animation.

To facilitate explanation, a general background of protein folding in biochemistry is presented. Terms appearing in subsequent sections will be defined briefly in this section. Following a summary of previous work in the area of molecular graphics, the problems associated with current molecular animation systems will be addressed. Based on observation of these problems, fundamental concepts which differentiate our prototype system from other conventional approaches are explained. Subsequently, design topics involved in the implementation of the prototype system are depicted. Finally, some of the frames produced from our prototype system are illustrated as experimental results.

Protein Geometry

Biochemistry is the study of the molecular basis of life. For instance, the discovery of the double-helical structure of deoxyribonucleic acid (DNA), the elucidation of the flow of information from gene to protein, the unraveling of the energy-conversion mechanisms, the determination of the three-dimensional structure and mechanisms of action of many protein molecules, are some of the outstanding achievements of biochemistry.

In virtually all biological processes, proteins play crucial roles in enzymatic catalysis, transport and storage of ions and small molecules, coordinated motion in a muscle, mechanical support of skin and bone, immune protection, generation and transmission of nerve impulses and control of growth and differentiation.

Amino acids are the basic structural units of proteins. An α-amino acid consists of an amino group, a carboxyl group, a hydrogen atom and a distinctive R group bonded to an α-carbon atom. An R group is called a *side chain* and various amino acids result from the variation of its component atoms. In proteins, the α-carboxyl group of one amino acid is joined to the α-amino group of another amino acid by a *peptide bond*. Figure 5-1 illustrates the formation of a peptide bond. Residues 1 and 2 can be repeated to form a *polypeptide* chain. An amino acid unit in a polypeptide is called a *residue*. By convention, the amino group end, the *N terminus*, is taken to be the beginning of a polypeptide chain. A polypeptide chain consists of a regularly repeating part, called the *main chain* (repetition of nitrogen, alpha carbon, carbonyl carbon), and a variable part, comprising the side chain or chains. The main chain is sometimes termed the *backbone*.

A remarkable characteristic of proteins is that they have well-defined three-dimensional structures, the rigid and planar peptide unit Figure 5-2 shows a pair of peptide planes. The hydrogen of the substituted amino group is almost always opposite the oxygen of the carbonyl group No freedom of rotation about the bond between the (carbonyl) carbon and the nitrogen atom of the peptide unit exists, since this peptide linkage has a partial double-bond character In contrast, the link between the alpha carbon atom and the carbonyl carbon atom is a pure single bond, and the bond between the alpha carbon atom and the peptide nitrogen atom is also a pure single bond Therefore, there is a large degree of rotational freedom about these bonds on either side of the polypeptide unit There is, relative to the peptide bond, free rotation about the bond between the alpha carbon and nitrogen atoms. The same is true for the alpha carbon to carbonyl carbon bond The rotational angles about these bonds are designated phi and psi, respectively The conformation of the main chain of the polypeptide is determined when the phi and psi angles for each amino acid are defined

As a result of this freedom, some three-dimensional patterns in protein structure can be found Alternately these patterns are called *secondary structures* as a conceptualization of a sequence of residues The *alpha helix* is a rodlike structure As shown in Figure 5-3, the inner part of the rod is made up of tightly coiled polypeptide backbones, while the side chains extend outward What stabilizes this structure is the hydrogen bond The alpha helix is stabilized by hydrogen bond between the NH and the CO groups of the backbone. The CO group of each amino acid is hydrogen bonded to the NH group of the amino acid that is located four residues ahead in the linear sequence.

The β *(pleated) sheet* differs markedly from the α helix in that it is a sheet rather than a rod Figure 5-4 illustrates a β sheet structure A polypeptide chain in the β sheet is fully extended, rather than being tightly coiled as in the α helix. Some other secondary structures [Ghé82, Ric81] such as β-turns, which are required for the directional change of the β-strand shown in Figure 5-4 , or coils, which form a short directional change from the a-helix to other structures, can also be found in nature G N Ramachandran and V Sasisekharan at the University of Madras, India, showed that unfavorable interactions between beta-carbon atoms of amino acid side chains with atoms of the polypeptide backbone make only three repeated conformations favorable: the right-handed alpha helix, beta strands and the left-handed triple collagen helix Alternations produce turns, loops and random coils

Protein engineers also define the tertiary structure [Str81] of molecules using the secondary structure as a basic building block. They expand the secondary structure into a pattern called the *domain* [Ric81], where alpha helices and beta sheets form various shapes such as barrels by parallel and anti-parallel combinations In this respect, the recognition of backbone shape is essential for the recognition of a higher level structure

In an alphe helix, each residue is rotated 100 degrees relative to the previous one, and is translated along the axis by about 1 5 angstroms (each turn containing 3.6 residues) Every carbonyl oxygen points upward and receives a hydrogen bond from a downward-pointing amide nitrogen of the fourth residue away from it. In a beta pleated sheet, the polypeptide backbone is relatively extended and each residue is rotated 180 degrees with respect to the previous residue The strands pack side by side, forming sheets in which each amide nitrogen donates a hydrogen back to a carbonyl oxygen in the adjacent strand and each carbonyl oxygen receives such a bond Acceptable phi and psi bond angles which correspond to these conformations can be determined by consulting a Ramachandran plot.

Polypeptide chains interact with their environments to fold up into discrete, highly organized and tightly packed three-dimensional structures Diverse overall shapes confer on proteins their powers of recognition and selectivity, catalysis, and specificity as structure-forming elements. Determining the conformation of proteins is therefore critical to understanding their roles in biological systems and biochemical pathways Most of the three-dimensional structural data obtained for biological macromolecules comes from X-ray diffraction experiments or the electron microscope. X-ray diffraction analyses require that molecules be ordered into fibers, sheets or crystalline arrays; however, not all macromolecules crystalize to form such ordered structures Electron microscopes resolve to about five angstroms, which is less than the accuracy required to characterize these structures A simple method for predicting how a protein will fold is clearly needed

The initial breakthrough in the study of protein folding occurred in 1957 when Christian B Anfinsen unfolded and refolded purified ribonuclease in the absence of cellular components This study established that all of the information needed to determine the final conformation of a protein can reside in the polypeptide itself How the sequence of residues determines the folding of proteins, and hence their conformations, has remained a major unsolved question in biochemistry

The term *conformation* in general encompasses the spatial arrangement of a molecule as determined by rotations about the single bonds In the specific case of the polypeptide chain, conformation describes the overall spatial organization The description of polypeptide conformation involves the specification of bond length, bond angles and angles of internal rotation about the sigma bonds

At the beginning of the life of the chain, residues in the sequence interact with each other and the solvent to cause the chain to fold up These interactions are almost solely dependent on the nature of the side chains which characterize the twenty amino acids Charged residues, such as aspartic acid, glutamic acid, and tyrosine, are located preferentially at the protein surface, where they can interact with water; residues in the interior are closely packed, with the solvent relatively excluded. Burying of the hydrophobic groups and surfaces is a major source of the protein stabilization For hydrophilic amino acids, interactions include both ion pairs and hydrogen bonds Hydrophobic amino acids are probably critical in the forming of solvent-excluding protein interiors. Within proteins, the conformations of the amino acid side chains are not strained, but represent low energy states

Recent progress on polypeptide chain folding has come from work in three areas:

1. Investigations of the conformation of folding intermediates using hydrogen exchange and nuclear magnetic resonance (NMR) techniques.
2. Pinpointing the residues in polypeptide chains that carry the information that determines the folding pathway by the use of polypeptide chains with amino acid substitutions at known positions
3. Analysis of steps within cells between the nascent polypeptide chain polymerized sequentially on the ribosome and the appearance of the native protein. For exported proteins, there is a complex series of

interactions of the nascent chain with cellular components, including helper proteins called molecular chaperones.

All three approaches are used to further elucidate the possible folding pathways.

Aside from the characterization of the role of the amino acids in protein folding, there have been computer-based approaches in determining the folding intermediates, and thus the folding pathways Energy minimization is a predominant simulation technique to determine a reaction pathway, since large movements which involve concerted displacements of many atoms or those requiring the crossing of energy barriers, cannot be investigated with molecular dynamics [Mel77, Str81]

In [Chu91, Jou91], energy minimization is used extensively as a simulation tool for the inspection of the conformational change occurring in large molecules. Bile salts are derived from cholesterol and their formation is the principal route of cholesterol degradation and elimination. In [Jou91], cholic acid (a kind of bile salt) layers are built on a screen by stacking individual cholic acid molecules together Subsequent energy minimization did clarify the change in the size and shape and orientation of channels peculiar to the micelle In [Chu91], ubiquitin molecules are joined together and their conformational change investigated. Depending upon the joint positions, various patterned structures were derived so that one could assume that a regulatory function was ruling the conformation Since energy minimization deforms the protein conformation in the direction of a more stabilized form, it is reasonable to assume that the folding process can best be simulated with it.

Therefore, the output conformations from energy minimization calculations can act directly as input frames in the animation However, the mere display of sequential frames from energy minimization does not result in smooth conformational change The degree of conformational change from one frame to another varies widely For instance, in one frame, some portion of a molecule may deviate from the previous frame by a wide degree of displacement while in next frame, the same portion remains relatively fixed Energy minimization generates conformational data relying solely on energy levels, and spatial distance in the conformational data cannot be reflected in its calculation

Literature Review

Molecular Graphics

Since Cyrus Levinthal's first attempt to draw molecules on a computer in 1966 [Lev66], molecular graphics as a field of computer graphics has undergone extensive research and improvement Traditionally, costly and tedious mechanical reconstruction of molecules in limited laboratory space has been the impetus for the preference for computer representation In addition, computer representation permits rotation and translation of a three-dimensional representation to aid in the recognition and understanding of a three-dimensional molecular structure [Sch79].

Molecular graphics is an ever-expanding field with immediate practical impact on the design and prediction of structures for useful, pharmacologically-active proteins Recent sophisticated demands require the control of specific protein conformations such as bond lengths, bond angles, molecule surface areas, molecular volumes, or boolean operation on areas and volumes with a simple interface device such as a mouse [Fle86, Piq86].

Individual atoms and the bonding relations between atoms fall into a few general classes in the representation techniques which have been applied to date. Among these computer graphical representations, the skeletal model, ball-and-stick model and space-filling model have their mechanical counterparts implemented by truncated plastic materials

The *skeletal model (or wireframe model)* [Sch79] is based on Kendrew-Watson's physical model of a molecule, typically with a scale of 1 Angstrom unit being equal to 20 mm and with a wire diameter of 2mm. This model shows only the molecular framework, as a collection of lines joining the atoms Junctions of bonds and their termini imply atomic position. Advantages of this model are simplicity and the capability of rapid imaging of large molecules (e g , those molecules where the number of atoms exceeds 1,000) The skeletal model permits the viewer to determine bond lengths, angles and dihedral angles [Sch79, Str81, Che88] with ease.

Devised by Corey, Pauling, and Koltun, the *CPK model (space-filling model or spherical model)* illustrates molecular surface shapes, since surface shapes determine such crucial biological functions as substrate-effector interactions [Sch79] While an atom is a quantum mechanical unit whose volume is specified by a probability distribution, experiments show that spheres can approximate atomic volume and shape [Che88] The van der Waals contact distance [Met77] determines the atomic radii in the CPK model This distance, defined as the length at which a repulsive force begins to exert influence when two atoms are drawn to each other, is largest among various bonds For non-bonded instances such as binding of sterically complementary atoms, van der Waals interaction as caused by temporal asymmetry in electron charge attracts atoms together Therefore, the CPK model treats the van der Waals radius as the size of an individual atom and expresses the molecule as the union of spheres [Max83, Met77] Nevertheless, the bonds between atoms are invisible in the spherical model. Because the bonds are the overlapping region of the atoms electron densities in the spherical model, they are buried under spherical surfaces and the relative geometry between the atoms is hard to distinguish

The *dot sphere model* uses dots to depict atomic surfaces or surfaces of a constant electron density The *ball-and-stick model* reduces the size of an atom relative to bond length to make the inter-atomic bonds visible. Varying the radius of the connection link in the ball-and-stick model imparts a sense of relative depth between two atoms as in a perspective view transformation Stereographic view pairs of this model [Syb89] allows depth perception of intersecting lines.

The interaction between the protein surface and a water molecule offers another modeling basis The *solvent accessible surface model* [Con83, Lan81, Max88], as a variation of the spherical model, makes it easy to visualize the enzymology of hydrophobic and hydrophilic reactions This model displays Van der Waals surface areas of an atom that is accessible to the spherical surfaces of a water molecule

With the growing popularity of the CPK model, various efforts in computer graphics area have been aimed at reducing time constraints involved in its use.

Polygonal approximation of a sphere uses a constant surface normal vector to speed up hidden surface removal [Rog85, New79] To avoid Mach effects on polygonal facets, the number of facets must be increased enormously Based on the scaling of color, precalculation of the shading function for a sphere [Sta78] was applied on a scan line basis to reduce the Mach effects by elimination of polynomial approximation. This sphere-shading algorithm is implemented with a dedicated display buffer [Sta86] to speed up the rendering of spheres in large molecules. In an attempt to reduce the complexity and thereby to reduce the time constraints required to draw intersecting spheres, a model of molecular structures with non-intersecting spheres has also been investigated [Sta88].

Rather than subdividing a sphere into a collection of polygons, Porter [Por78] treated a sphere as a primitive and exploited the fact that a sphere projects into a circle regardless of the position of viewpoint. Porter applied Bresenham's image space circle generation algorithm [Bres77] to avoid square-root calculations. However, this method suffers from aliasing [Cro77, Cro81] by showing jagged edges. Such aliasing effects can be greatly reduced by the use of post-aliasing lookup tables [Wan89]. If one simplifies a sphere into a circle [Eva78, Max78, Max83, Kno81], a conventional hidden line removal method can be used to calculate visible arcs. In this method, a circle or filled circle will approximate a sphere, but the model lacks visual reality. In the case of non-overlapping spheres, an object space pruning algorithm [Fra81] is possible. By excluding the surfaces which are declared hidden in previous removal stages, the algorithm maintains the complexity linearly proportional to the number of spherical objects.

Equally as important as the spherical model is the ribbon model, which has been used to visualize a higher level of conceptualization, especially in protein engineering. Because the amount of graphical information in a globular protein is very large, researchers tend to lose interest if they are confronted with the full details of the individual atoms represented by a spherical model. Rather, they need to interpret local distribution of atoms in terms of global secondary structures, which are regular arrangements of the sequence of residues. For instance, biochemists are interested in whether they can find a particular amino acid [Jas88] in a particular location on a long residue sequence. In this respect, the ribbon model, as a framework for the recognition of higher level structures, such as the tertiary structures, plays an important role.

Although protein engineers may fit together several side chains during the design process, their ultimate interest lies in the backbone conformations, since the stability of the molecule is determined by the spatial pattern of such conformations. The ribbon model [Sch79, Str81] in this context portrays a molecule as multiple threads approximating the lines joining backbone atoms. Although the backbone atoms are defined to include all the nitrogen, alpha carbon and (carbonyl) carbon atoms, the line joining the adjacent alpha carbons has been treated as the backbone line alternatively. Figure 5-5 illustrates the alternative backbone line. Figure 5-6 illustrates a ribbon model representation of various conformational structures. Instead of a single thread, multiple threads are drawn in parallel to help the viewers perceive three-dimensional depth on a two-dimensional screen. Various patterns of naturally stable secondary structures such as helix, sheet, coil and turn can be visualized with ease using the ribbon model.

Overall, the exploration of molecular graphics to date has mainly concentrated on a static display of a molecule and the use of models varied to suit the diverse needs of the protein engineers. Depending on what aspect of a molecule should be emphasized and visualized, various models have been proposed. Nevertheless, the modeling aspects with regard to the animation of molecules have not yet been probed. In the following section, we will examine and analyze how previous molecular animation systems have been developed.

Molecular Animation

With the growing interest in molecular graphics, the obvious extension of existing means of computer representation is an animated graphic display of the molecular motion. Feldmann and Levitt [Fel80], in a 1980 film of the molecular dynamics of bovine pancreatic trypsin inhibitor (BPTI), demonstrated the power of computer animation as a tool for the researcher. Fundamental concepts were easy to grasp using visual effects involving motion. The technique of computer animation shows promise in applications from non-invasive robotic surgery to the prediction of structure for enzymes, and to the animated display of such molecular processes as DNA unwinding or protein folding/unfolding equilibria.

In the decade since these earliest efforts at computer animation of the motions of a molecule, the computer-aided design, manipulation and display of molecular structures have grown into a one hundred million dollar-a-year business. However, most of the existing molecular graphics packages put great emphasis on enhancing the photorealism of a static view of molecules, while ignoring the potential for animated display of molecules.

Those graphics systems which do attempt the animation of molecular dynamics data to simulate molecular motion share a number of common characteristics as follows:

Foremost of these is that they require the use of a mainframe computer, since the generation of realistic images can be prohibitively expensive. Because the computer representation of a spherical or space-filling model requires an enormous amount of CPU time, the simpler wireframe model has proven more useful in all attempts to date at molecular animation. The wireframe model as an animation tool [Del84, Fuj86, Tod83] permits rapid imaging of large molecules, but has the disadvantage of such distracting visual effects as flipping side chains, abrupt changes in angles and disjointed motion. As a result, viewers are easily distracted from the global changes in conformation. Second, conventional concepts of animation as a series of sequential displays of a frame have proven of limited use when applied to molecular dynamics. Third, most of the approaches to date have concentrated on increasing interactivity [Del84, Fuj86, Tod83], so that a user could control animation speed and viewpoint, and freeze the animation stream in order to analyze individual conformations. Such controllability and the added capability to analyze a static scene are meaningful in that they help recognize the three-dimensional structure of the molecules, but are useful only if a proper model for animation is employed.

To summarize the problem, currently available animation systems such as the Winchester Graphic System [Tod83] or BRUGEL [Del84] have shown too much emphasis on the implementation of the controllabilities. The molecular model predominant in the previous animation system was the wireframe model, which intrinsically produced such visual anomalies as high-frequency motion of flipping side chains and abrupt motion discontinuities. Essentially, the wire frame model is not suitable for animation. In addition, the conventional systems could not show smooth movements, since a series of folding intermediates, as is, is displayed without any consideration of the possibility of generating inbetween frames.

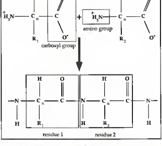

Figure 5-1. Peptide bond between residues. The atom types are, H for hydrogen, C for carbon, O for oxygen, N for nitrogen, Cα for alpha carbon. The R group is a collection of atoms which characterize the given residue.

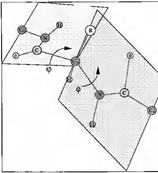

Figure 5-2 Definition of φ and ϕ φ refers to rotation about the C_α-C single bond; ϕ refers to rotations about the C_α-N single bond In a fully stretched-out polypeptide chain, $\phi = \varphi = 180$ degrees

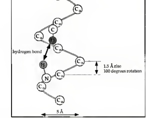

Figure 5-3. A schematic diagram of a right-handed α helix. Hydrogen bonds between NH and CO groups stabilize the helix. In the α helix, the CO group of residue *n* is the hydrogen bonded to the NH group of residue (*n* + 4)

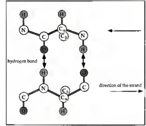

Figure 5-4 A schematic diagram of the anti-parallel β pleated sheet. Adjacent strands run in opposite directions. Hydrogen bonds between NH and CO groups of adjacent strands stabilize the structure. The side chains (Cβ) are above and below the plane of the sheet

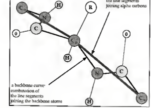

Figure 5-5 A backbone conformation Atom types are, N nitrogen, C carbon, O: oxygen, H. hydrogen C_α alpha carbon. The R group represents a collection of atoms which characterize a given residue.

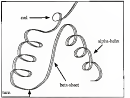

Figure 5-6 A schematic diagram of the protein secondary structures represented in the ribbon model A coil is relatively tightly wound compared with a turn

System Overview

Two specific aspects of our animation system should greatly improve the quality of the conventional molecular animation systems These are,

i) Modified use of the ribbon model

ii) Introduction of key-frame inbetweening

The G^1 free form spline developed in Chapter 3 makes up the theoretical foundation for the modified ribbon model which will be discussed subsequently The method to control the motion speed developed in Chapter 4 will be used in key-frame inbetweening or our animation system

The Solid backbone model

Although the ribbon model has been used to generate a static view of protein conformation, the disadvantages of the model, when used as an animation tool, can be summarized as follows

1. When animated, the multiple parallel threads making up the ribbon results in a flipping motion which distracts viewer's attention from the conformational change of the backbone
2. The original purpose of drawing multiple threads in the ribbon model was to enhance depth perception in a static view But the threads enhance depth perception only when a viewer has enough time to examine the width and twist of the ribbon. In a dynamic display of frames, however, the time between the frames is very limited since the frames should be displayed at a rate of about thirty frames per second in order to produce the visual illusion of animation.

3. In the ribbon model, the curve representing a sequence of the backbone atoms does not necessarily pass through the backbone atoms themselves

In order to resolve these problems involved in the animation of ribbon model, we define the *solid backbone model* as a proper tool for the animation as follows:

1. The solid backbone model is a combination of cylinders with their axes joining the backbone atoms. Thus the backbone curve appears similar to a tube or a cable.
2. The solid backbone model passes through the backbone atoms exactly.

Because it is defined as a three-dimensional solid (i e , combination of the cylinders), the flipping motion of the multiple threads during the animation can be removed Meanwhile, the three-dimensional depth of the protein conformation can be perceived through the shading on the surface of the cylinder The shading is a more natural way to recognize a three-dimensional structure and it allows immediate recognition of the backbone conformation during animation. In a sense, the model is also suitable for the static view of the backbone conformation, since the only reason the ribbon model is used in the representation of the protein backbone is for depth perception.

Various illumination techniques for shading can be found in [Fol82, Rog85] Usually, the illumination models are applied to a polygonal surface using a constant normal of the polygon. In the solid backbone model, a cylinder is made up of patches of small polygons and the number of polygons approximating the cylinder can be chosen arbitrarily Although the image of a cylinder becomes close to a real cylinder as the number of the polygons increases, the computation involved in the calculation of the intensity of a pixel becomes costly also. In animation, however, it is generally believed that a rough shading will suffice to represent an object, since viewers tend to ignore the minute details of the shading while the image changes.

Another technique involved in improving the three-dimensional perception of a solid backbone model is the use of hidden surface removal algorithms These algorithms attempt to determine the lines or surfaces that are visible or invisible to an observer located at a specific point in space The fundamental assumption made in these algorithms is that, the farther an object is from the viewpoint, the more likely the object is to be totally or partially obscured by one closer to the viewpoint

Various solutions, such as a scan line z-buffer algorithm [Ath83, Sta78], visible surface ray tracing [Whi80, Gla89], floating horizon algorithm [Wan89 Wri73] can be used for the back face culling of the solid backbone model. Note that this process was not used in the ribbon model, since the the ribbon threads were simply overlaid and intersected regardless of their distance from a viewpoint and the viewers had to examine the twist and width of the threads to determine which portion of the backbone curve is closer to them

One of the most important rationales behind our animation system is that it also leaves room for the display of the previous molecular models As mentioned earlier, the spherical model itself is not animated in our system because protein engineers are more concerned about the conformational change represented by the backbones, which are usually hidden under Van der Waals surfaces in the spherical model representation

However, the importance of the spherical model must not be overlooked since the geometry of spherical surfaces determines some of the crucial properties of the molecules. Often protein engineers need to stop the animation flow and query about the geometrical information of the atoms, in a static view of a scene. In such a situation, the spherical model can be a useful tool for the analysis and this is the main reason why we make the solid backbone model pass through the backbone atoms.

By making the backbone curve pass through the backbone atoms, viewers can locate the exact position of backbone atoms. Therefore, they may notify the animation system of their area of concern by clicking on the backbone atoms at the concerned area. The animation system, in turn, can display the spherical model of the backbone atoms in that specific area. Furthermore, entire atoms belonging to clicked residues can be displayed, provided with a proper data structure [Aho83, Sam90] that could link the positional information of the atoms within a residue. Because alpha carbon joins nearby nitrogen and carbon in the same plane, specification of a sequence of backbone atoms (or alpha carbons) can completely recover the structure of all atoms related to those backbone atoms with proper setup of a database [Dat86, Elm89]. This way, only those atoms which interest the viewer, which are hidden inside most of the time in the spherical representation of an entire molecule, can be viewed in detail.

In exchange for the pleasing visual appearance of approximating splines, which do not necessarily pass through the backbone atoms, the accuracy of the backbone curve can be enhanced. At least, backbone atoms lie on the backbone curve. Even if the backbone curve is made up of alpha carbons only, there is a greater chance that the nearby nitrogens and carbons are close to the backbone curve than is the case with the approximating splines. The approximating splines are best suited for the display of secondary structure patterns which are usually very stable in nature and thus tend to appear at the final stage of the folding process. Since the folding process in general is the transition from the random, unfolded conformation to the secondary structural patterns, intermediate states do not generally exhibit such patterned structures. Therefore the conformation in transition does not have to be visually pleasing. Instead, the graphical representation should be focused on representing a more accurately the shape of the folding intermediates.

In practice, the property of passing through the backbone atoms acts as a bridge between the static and the dynamic display (i.e., animation). As long as there is a proper tool for linking the dynamic and static display, various static models presented previously can be exploited and incorporated into the animation system. At the instant the animation is interrupted for a static analysis, the system may go back to the nearest key frame and the static display can be started with the position of the alpha carbons on that key frame. Therefore, such static display can be imbedded as part of the dynamic display with this scheme.

Figure 5-7 clarifies this point. In the upper left, a dot sphere model of a certain protein residue sequence is drawn. The positions of the alpha carbon atoms are indicated by the joints of the straight line segments and the joints themselves are the centers of the dot spheres representing the alpha carbons. A smooth curve running along the dot spheres is an example of the approximating spline. Notice how the spline misses the centers of the alpha carbon atoms. The backbone curve for the situation at hand leaves the position of the backbone atoms by a large margin. Therefore it is hard to mark the position of backbone atoms on such a curve. The top right figure is a spherical model of the same molecule, and the bottom figure is a zoomed view of a portion of that molecule. By making the backbone curve pass through the backbone atoms, the position of the backbone atoms can be marked *on the curve*, and the alpha carbon locations can be completely informed to the animation system just by designating their positions If, on the other hand, the backbone curve is constructed by an approximating spline, there is no way to restore the exact position of the backbone atoms upon stopping the animation flow This is the main reason why the solid backbone model is set to pass through the backbone atoms.

Frame Inbetweening

The second aspect of the proposed system is the incorporation of inbetweening techniques into the solid backbone model. The inbetweening technique as a research area of computer graphics has been investigated in Chapter 4 in detail and thus this section will present its application in the context of the animation of protein folding.

To date, none of the existing molecular graphic systems has attempted to use the inbetweening technique in the context of animation. This is true even for the simulation of the *protein dynamics* which can be loosely defined as conformational fluctuation over a given time span [Met77, Str81] Previous endeavors [Del84, Fuj86, Tod83] in molecular animation share some common characteristics, as follows:

1 In the modeling aspect, most of the systems use the wireframe model as an animation tool

2. Since a full bond is represented in the wireframe model, high frequency motions of sidechains could distract the viewer s attention

3 Even if only the chains connecting the backbone atoms are displayed, the motion transition is inherently abrupt, since the joint angles of the wire frame can vary widely from one frame to another

By incorporating a smooth curve in the solid backbone model, the problems associated with the first two characteristics of the wireframe model could be removed. However, the last characteristic should be handled properly to assure enhanced display of the protein folding process.

It is important to note that the mere collection of intermediate backbone shapes is not sufficient for animated display, since the computer animation technique relies heavily on visual illusion. When a collection of frames representing the backbone is displayed sequentially, the visual illusion of animation is possible only when a sufficient number of static frames, which are similar in shape, are provided. If the image of the current frame deviates remarkably from that of the previous frame, the previous image does not help animate the scene, although it remains and becomes overlapped by the current frame visually.

In view of the preparation for the shapes of the folding intermediates, it can be said that visual resemblance between adjacent intermediates cannot be assured. Because of the limited technologies [Kim82], to date, only a few folding intermediates have been able to be captured and displayed. Theoretical postulation on the folding pathway [Cho84, Wri88] does not necessarily document minute transitions from one frame to another.

Rather, a combination of the secondary structures, which belong to a very limited pattern in terms of the entire conformational change that folding process undergoes, is widely used in the explanation of the folding pathway. For instance, in Figure 5-8, a formation of polypeptide chain is explained in terms of the formation of a transient alpha helix.

Although a theoretical abstraction can be described by such a schematic diagram, the animation of the folding transition involving a detailed conformational structure cannot be presented in such a discrete way. The same holds true for calculation of the protein dynamics or energy minimization. For instance, if energy minimization is set to stop and produce the conformational data for a predefined small change in the energy level, the conformation thus produced can be varied greatly

No matter what folding pathway is assumed, the acquired folding intermediates should be treated as key frames to exploit the visual illusion of animation using computer graphics technology

Inbetweening frames has proven to be a useful technique to compute an intermediary point by interpolating between two key drawings in animation As was discusted in the Chapter 4, it smooths out the adjacent key frames by inserting extra frames between them so that it assures smooth motion transition

By filling in a pair of distinct key frames with the inbetween frames, the resemblance between frames is exploited, so that the visual illusion leading to animation may take place Given the conformation for a certain instant and the conformation at a later time, inbetweening is the process of generating intermediary shapes Figure 5-9 illustrates how the backbone curves change their shapes gradually by the incorporation of the inbetweening technique Pairs of key frames (i.e , two folding intermediates of a protein structure) produced from various sources are imported The sources can be a postulated folding intermediates, protein dynamic calculations, measured folding intermediates, or the conformation resulting from energy minimization During playback of this sequence, the Interpolating scheme computes those frames between key frames [Ree81, Smi83] and displays them at the cine rate on the monitor This method contrasts with some of the previous approaches that specify a single frame made up of a single protein structure and command it to move by a certain rule [Bur71].

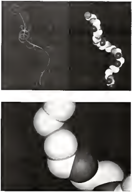

Figure 5-7. An approximating spline used for the estimation of the backbone curve

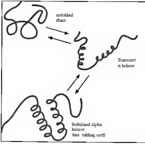

Figure 5-8 A schematic diagram showing the transition between folding states Two segments of an unfolded polypeptide chain transiently become an α helical These helices are then stabilized by the formation of a complex between the two segments

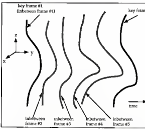

Figure 5-9 A schematic diagram of a folding animation. Frame numbers are assigned in time sequences. Starting key frame (key frame #1) is treated as the first inbetween frame for reasons described in Chapter 4. A total of five inbetween frames are produced to smooth out the transition between the two key frames.

System Design

This section describes a prototype system for folding animation. The solid backbone model described in the preceding section will be implemented and used as a static frame representing a folding intermediate. The inbetweening techniques will be incorporated to produce extra frames which would enable smooth transition between two folding intermediates. In making up a solid backbone model, the G^1 free form splines developed in Chapter 3 will be used to enable wiggle-free representation of the backbone curve. The same spline will be used in producing a wiggle-free motion path of an individual atom belonging to a folding intermediate over the given time span. The speed control techniques developed in Chapter 4 will be used to generate a constant speed during the animation. Generated frames will be illustrated in the next section.

The prototype system is made up of a combination of hardwares: VAX 11/780, Evans/Sutherland PS/390 and a Macintosh IIx with 8 megabytes of RAM and 120 megabytes of hard disk space. SYBYL, a molecular modeling software from Tripos Inc., was used for the preparation of the input data in our animation system.

Figure 5-10 shows a block diagram of the prototype system including the input preparation stage. The role of each block is as follows.

1. Input preparation

In the prototype system, we concentrate on the folding animation of a protein made up of twelve amino acid residues. Using Evans/Sutherland

PS/390, a sequence of residues of a ubiquitin molecule which belongs to the alpha helical portion is extracted. It is:

ILE1-GLU2-ASN3-VAL4-LYS5-ALA6-
LYS7-ILE8-GLN9-ASP10-LYS11-GLY12.

where

ILE = isoleucine, GLU = glutamic acid, ASN = asparagine,
VAL = valine, LYS = lysine, ALA = alanine,
GLN = glutamine, ASP = aspartic acid, GLY = glycine

The principal impetus behind the prototype system, in terms of biochemical interest, is twofold. First, if we design and construct a protein by connecting sequences of such extracted residues in a geometrically linear fashion, can energy minimization yield the same conformation (an alpha helix) as was imbedded in the original ubiquitin molecule? Second, is there any role of a specific site that characterizes the folding process of the designed molecule? The animation may suggest an answer to the second question.

The VAX 11/780 was used to calculate the gradual, conformational change of the geometrically linear sequence of residues, upon energy minimization. In each run, the resulting conformational data with a certain energy level makes up a key frame to be used in the next stage A total of fifty key frames were generated with successive reduction in the energy level. The output file format, MOL2 file format [Syb89], is a collection of a three-dimensional Cartesian coordinates of all the atoms including the backbone atoms

2- *Backbone scanner*

The reason why the backbone scanner is separated from the input preparation stage is to allow more flexibility of the system so that various formats of conformational data can be handled. Atomic coordinates from different sources and different energy minimization schemes can be incorporated into our system with a proper modification of the backbone scanner. The scanner extracts either the location of alpha carbons or can be set to extract entire backbone atoms such as nitrogens, carbons and alpha carbons from the given conformational output file of the input stage.

3. *Endpoint generator*

Generally, in the cubic spline method, four data positions are needed to evaluate a curve interpolating two inner data points. Two exterior point positions are required to calculate and maintain the tangential continuity as was discussed Chapter 3. This is true both for the C^1 and G^1 class splines. In terms of the data points made up of the twelve residue sequences, positions of residues #3 and #6 are used to generate a backbone curve interpolating between the residues #4 and #5. Therefore, inside the sequential data points, this restriction does not cause any problems. However, to evaluate an interpolating spline running from residues #1 and #2, an extra data point preceding residue #1 is needed and this data point somehow affects the curve shape in the interpolated region. The same holds true for the evaluation of the curve running from residues #11 and #12. Thus, two extra data point locations should be supplied as an end condition.

Although some effort has been made to produce a possible end condition [Bart87, Far90], an additional complexity of the calculation is involved in the evaluation of the endpoints. For instance, a clamped end condition requires the prescription of tangent vectors at the data points and the quadratic end condition requires evaluation of the second order derivatives. However, no generally-known best solution to the end condition problem exists and it is reasonable to assume a number of possible positions of the endpoints depending upon the situation encountered In this prototype system, the endpoint preceding residue #1 is assumed to be in the opposite position of residue #2 with respect to residue #1, and similarly for the endpoint following residue #12.

Figure 5-11 shows a data structure representing the positional information of the frames. The block composed of white squares represents the original data received from the input preparation stage. Each column of the block represents the position of alpha carbons making up a backbone conformation in space For instance, the square labeled 1 is the three-dimensional Cartesian coordinates of the position of the alpha carbon atom belonging to the residue #1, isoleucine. Therefore, starting from the left, each column can be regarded as a sequence of key frames over the time interval and the total number of columns is 50, as was determined by the input preparation stage

While each column of Figure 5-11 shows a backbone conformation at a certain instant, a row in the figure represents the positional change of an alpha carbon over a given time interval For instance, the first row of the block of white squares specifies the spatial locus of the alpha carbon atom of the isoleucine as a function of time

The endpoint generator works in two ways in the figure First, an additional imaginary residue position is attached to both ends of a column as shown in the figure, thereby making up each frame of 14 residues These additional residues will be used as the endpoints for the generation of the backbone conformation in the spline interpolation module so that the interpolated curve may range from residues #1 to #12 in its entirety Second a pair of additional endpoints is generated for each row of the figure. These endpoints are required to evaluate the motion path of a single atom in the time domain. Just as the endpoints are needed for drawing the static beckbone curve at a certain time, so are they needed for drawing the motion path of a single atom over a time interval Row-wise addition of the extra atomic positions will generate two extra key frames, one preceding the first key frame, the other following the last key frame. The additional key frames will be used for the interpolation of the motion path in the subsequent spline interpolation module

4 Row/column converter

This routine converts the collection of columns in Figure 5-11 into rows and vice versa Since the first step of our spline interpolator is the interpolation of motion path in the time domain, and since the interpolation is applied for each atom, the column data should be converted into row data Therefore, fourteen files, each containing the locus of each residue, including the two extra endpoints, will be generated as a result of running the converter

5 Spline interpolation module

The Incremental Knot Spacing method developed in Chapter 4, and the G^1 free form splines developed in Chapter 3 are applied to the motion speed interpolant and the backbone shape interpolant, respectively, in this module. Although two separate routines make up this module, they share a common routine in practice, since both adopt the G^1 free form splines as their interpolation scheme The same interpolation method can be used both for the generation of the motion path of a single atom over a time interval, and the generation of a backbone curve at a certain instant. Consequently, the timing and the shape of inbetween frames are determined by this module.

The main reason why they are separately treated lies in the fact that the additional flexibility of controlling motion speed is required in the motion speed interpolant. Our prototype system is made to maintain constant speed between pairs of key frames, but one might need to slow down the change of the conformation in a certain key frame range and this is possible by varying motion speed parameters in the motion path interpolant routine. By the same token, the number of points approximating a backbone curve can be arbitrarily chosen in the backbone shape interpolant. For instance, if a curve between two residues is approximated by the combination of the line segments joining eight internal points, the curve can be said to be more accurately approximated than one constructed by joining four internal points. Since the number of residues representing a large molecule could greatly be increased, there should be a trade off between the number of internal data points and the computation time involved.

As a result of running the motion path interpolant, additional inbetween atomic positions are generated for each row of Figure 5-11. For instance, if eight inbetween frames per pair of key frames are specified for the motion path interpolant, a total of 401 frames (8 * 50 + 1) are generated to enable smooth transition between backbone conformations. Since these interpolations run for each atom making up a row, and since the actual backbone conformation is built up upon the data points represented by columns, the output of the motion path interpolant should be fed into the row/column converter.

Finally, the backbone shape interpolant is used to generate the backbone curve corresponding to each of the 401 frames which are produced by the motion path interpolant

6 Translator

There is a particular aspect of motion dynamics to be considered in association with the animation of protein folding. If the frames illustrated in Figure 5-12 are displayed sequentially, as is, the global motion of the backbone curve will include the displacement of the backbone from one frame to another. The backbone curve, in its entirety, will be translated as if the backbone itself is moving. As a matter of fact, all the atoms involved in the energy minimization calculation are translated and the data given in the input preparation stage do contain the positional displacement.

Nevertheless, in the animation of the folding process, what we are interested in is the relative deformation of the inside conformation of the backbone curve and not the global translation of the curve. Suppose, for example, that the relative motion of the moon and the earth is to be examined by an animation. If the scene were captured in such a way that the sun, and the circular movement of the moon and the earth around it, is included, viewers would be overwhelmed by the global motion of the moon and the earth around the sun. Although the relative motion between the moon and the earth is correctly animated, it might go unnoticed. By the same token, the relative inside deformation of the backbone curve may be ignored if the global displacement of the backbone curves is also included in the animation

Therefore, a reference point should be postulated on the backbone curve to facilitate the perception of the relative deformation of that curve

Once a reference point is determined on a certain frame, all the residue locations in the other frames should be translated such that they all meet at the reference point. Figure 5-12 shows the difference in the three animated frames resulting from variation of the reference point. The reference point on the first illustration is set to the N-terminus, while the second one is set to the C-terminus of an amino acid residue. If a reference point is fixed near the middle of the backbone curve, the animation will appear such that the backbone deformation propagates with respect to the reference point. In the prototype system, the position of the first residue is used as the reference point. However, it is important to note that no matter what strategy is taken, the relative position of an atom within a backbone curve remains fixed, since all of the atoms are translated as a group, by the translator module.

7 Display module.

This module is responsible for the generation of the cylindrical surfaces running along the length of the backbone curve and there are some geometrical calculations involved in the estimation of a cylinder. In practice, a cylinder can be approximated by an arbitrary number of polygons as shown in Figure 5-13. In this figure, four polygonal planes parallel to the line segment joining the center of surfaces A and B are shown.

Suppose that the backbone curve shown in the figure is to be approximated by the line segments joining the four points $\mathbf{C}_1$ through $\mathbf{C}_4$. If we denote x, y, z components of $\mathbf{C}_i$ as

$$\mathbf{C}_i = (x_i,\ y_i,\ z_i),$$

then the surface normal vector of the surface A can be approximated by

$$n = ((x_3 - x_1),\ (y_3 - y_1),\ (z_3 - z_1))$$

Therefore, a vector which lies on the surface and is also perpendicular to the surface normal can arbitrarily be written as,

$$r = ((y_3 - y_1),\ -(x_3 - x_1),\ 0).$$

Given the radius R of the cylinder, a point P on the circle lying on the surface A can be calculated as,

$$P_x = x_2 + R(y_3 - y_1) / |r|$$
$$P_y = y_2 - R(x_3 - x_1) / |r|$$
$$P_z = z_2$$

where

P_x, P_y, P_z are x, y, z components of the point P

Having set a point on the circle, the rest of the points on that circle can be calculated easily by rotating the point with a certain angle with respect to the center of the circle However, in order to build a polygon, another point P on the surface B is required to match the point P on the surface A. A simple approach is taken in our prototype system The matching point is assumed to be the intersection of the surface B, and the line parallel to the vector $(C_3 - C_2)$ while passing through the point P

A perspective projection [Fol82, New79] was employed in the prototype system If a square surrounding the backbone curve in the x-y plane is assumed, the center of projection is made to be located at a certain distance from the center of the square with its direction parallel to the z-axis Since the object further away from the viewer appears smaller in the perspective projection, the method provides a depth cue, an indication of which portions of the image correspond to parts of the object which are close or far away

A simple illumination model [Fol82, Rog85] is used to shade the solid backbone model such that the Lambertian cosine term is added to a constant diffuse light. According to the Lambert's cosine law, the intensity of a light reflected from a perfect diffuser is proportional to the cosine of the angle between the light direction and the surface normal. In addition, the ambient light explains the light scattered from the surroundings. Therefore, the light intensity of a certain polygon is

$$I = I_a k_a + I_l k_d \cos \theta$$

where

I is the intensity of the light reflected from the polygon,

I_a is the incident ambient light intensity,

k_a is the ambient diffuse reflection constant,

I_l is the incident light intensity from the distant source,

k_d is the diffuse reflection constant,

θ is the angle between the light direction and the surface normal.

Since the light direction is assumed to be parallel in our system, and since the surface normal of a polygonal surface is constant, the reflected light intensity is constant for each polygon.

The display module is mainly concerned with the generation of static images of the individual frames. Various types of commercially available sequencers can simply organize and display the individual frames for animation. However, we limit our interest in this dissertation to the generation of sequential frames based on the the solid backbone model approaches and further investigation of the sequencers is not covered.

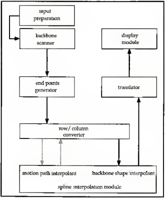

Figure 5-10 Block diagram of the prototype system

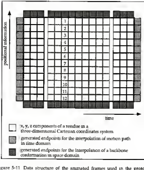
positional information
1
2
3
4
5
6
7
8
9
10
11
12
time
□ x, y, z components of a residue in a three-dimensional Cartesian coordinates system
▨ generated endpoints for the interpolation of motion path in time domain
■ generated endpoints for the interpolation of a backbone conformation in space domain

Figure 5-11 Data structure of the animated frames used in the prototype system. Numbers represent the residue sequence of the prototype system

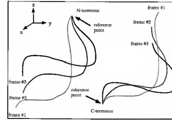

Figure 5-12. Referencing the frames at different places If the N terminus of the frame #1 is set to a reference point, all backbone atoms in the frame #2 and #3 are translated by the same amount as their N-termini are translated to the N-terminus of the frame #1

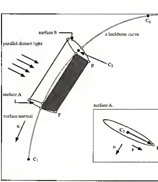

Figure 5-13. A cylinder approximated by four polygons. The vector **n** represents the surface normal of the surface A. The vector **r** lies on the surface A and is perpendicular to the vector **n**. The interpolated backbone curve is approximated by a combination of the line segments joining the points C_i. Point P and P' are two matching points on surface A and B, respectively.

Experimental Frames

Some of the frames produced by the prototype system are illustrated in this section as visual experiments using the solid backbone model discussed above

Figure 5-14 shows the redrawing of the ubiquitin molecule shown in Figure 1-1 Instead of multiple threads, four polygonal surfaces are used to approximate the cylindrical surface representing the backbone in the solid backbone model Although roughly approximated, shading and removal of the hidden surface is explicit enough to convey the spatial distribution of the backbone curve The top figure shows the B-spline interpolation of the backbone curve by a combination of solids and the bottom figure shows the solid backbone model drawing. Although the solid backbone model is defined only in terms of the interpolating splines, this example is produced for the purpose of making a comparison between the two As was shown, the B-spline is smoother than the free form spline and is visually more pleasing However, there appears to be more error in the B-spline approximation.

Figure 5-15 illustrates the alpha helical portion of the ubiquitin molecule shown in Figure 5-14 The helical portion ranges from residue 20 to residue 36 of the ubiquitin molecule The top figure is the B-spline approximation and the bottom figure is the solid backbone model approximation. With white dots representing the location of the alpha carbon atoms, the B-spline can be said to be wide of the mark. Although aesthetically pleasing, the backbone curve is fairly far from the actual backbone atoms In contrast, the solid backbone model drawing could achieve more accuracy by making the curve pass through all the positions of the alpha carbons There is a greater chance for the other backbone atoms, such as nitrogen and carbon, to exist in the vicinity of the curve Even if the smoothness is reduced somewhat compared with that of the B-spline, visual continuity is preserved in this model.

Figure 5-16 shows a pair of zoomed-in views of the part of the alpha helix in Figure 5-15 The difference in the intensity of reflecting light (i e , shading) can be shown clearly for individual polygonal patches Notice that the radius of the cylinder, and thus the sides of the polygon, is not regular because of the limited number of polygons used for the approximation. As the number of line segments representing the backbone curve increases, and as the number of polygons approximating a cylinder increases, such irregularity can be reduced.

Figure 5-17 is the array of the twelve residue sequences used as the input for energy minimization It is modeled simply by concatenating each residue in a linear fashion As a result of energy minimization 50 frames were available for our animation system. However, for purposes of the example, only the first three key frames are presented Figure 5-18 illustrates a solid backbone model drawing of the three key frames representing the folding transition of the protein. Notice that the conformational changes between adjacent frames are great enough to prevent visual illusions when they are directly displayed for an animation

Figures 5-19 and 5-20 illustrate the frames produced by the inbetweening technique The motion speed is made constant for each atom belonging to the backbone conformation Figure 5-19 shows the inbetween frames interpolating key frames #1 and #2, while Figure 5-20 shows those interpolating key frames #2 and #3 As can be seen, any two adjacent frames look quite similar, so that the visual illusion leading to the animation can happen if the frames are displayed sequentially In other words, accumulation of the small unnoticeable changes between adjacent frames makes the abrupt conformational changes of the key frames look smoother For instance, although the first and last frame of Figure 5-19 are highly dissimilar, the individual conformational change in any pair of adjacent frames in the figure is very slight This is the greatest advantage of the inbetweening technique.

Summary

As an application of the free form splines developed in Chapter 3, and the speed control strategies developed in Chapter 4, a prototype animation system for protein folding has been presented. In addition, a new graphical model suitable for molecular animation has been proposed and tested in this chapter The basic concepts involved in the solid beckbone model are the solid drawing of the backbone and introduction of the inbetweening technique. In addition, the advantage of the interpolating spline employed in the solid backbone model over an approximating spline has been described and illustrated.

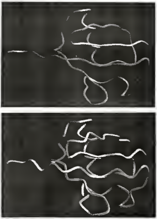

Figure 5-14 Comparison of the B-spline and free form splines The backbone of the ubiquitin molecule discussed in the Figure 1-1 is drawn in solid. Bottom figure is a representation of the solid backbone model and the top figure is a variation of the solid backbone model

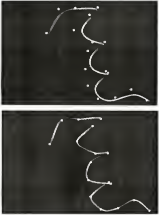

Figure 5-15 An alpha helix portion of the ubiquitin molecule The curve approximating the backbone of residue 20 through 36 is drawn White dots represent exact location of residues Top figure is the B-spline approximation of the backbone, while the bottom one is the solid backbone model used in our prototype system

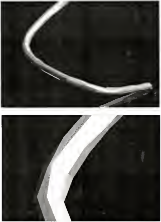

Figure 5-16 Zoomed view of a portion of the alpha helix in the Figure 5-15 Top and bottom figures are zoomed in by a factor of 4 and 16, respectively

Figure 5-17 The linear arrangement of a sequence of twelve residues used in the prototype system Top left is the beginning N-terminus and bottom right is the C-terminus

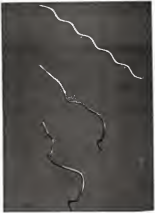

Figure 5-18 Three sequential backbone conformations used as the key frames in the prototype system These frames and the coordinates are taken from the energy minimization output

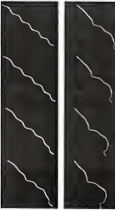

Figure 5-19. Six extra frames produced by the inbetweening technique. The second key frame is also shown for reference. The sequence is from top left to bottom left and from top right to bottom right. Notice the gradual change from one frame to another.

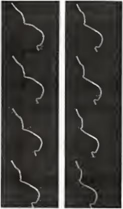

Figure 5-20 Six frames produced by inbetweening key frame #2 and #3 The third key frame is also shown for reference The sequence is from top left to bottom left and from top right to bottom right Notice the gradual change from one frame to another

CHAPTER 6
CONCLUSIONS

This dissertation is primarily concerned with the animation of the protein folding process in biochemistry The solid backbone model presented in this dissertation provides a convenient tool for the visualization of the main polypeptide chain in any animation environment

Most importantly, in the model, alpha carbons can be imbedded and marked as part of the beckbone curve. When the animation stream is stopped for further analysis of a static conformation, such marked positions can be used as an interface between system and viewer Portions of a large macromolecule can be seen in the spherical model by designating the alpha carbons belonging to the area of interest, using a device such as a mouse Provided with a proper database, the system can recognize the clicked location and completely recover the location of other atoms belonging to the peptide planes centered at the alpha carbon

As a result, those van der Waals surfaces buried under outside atomic surfaces can be exposed selectively This was not possible in an animation relying entirely on the spherical model, despite the enormous amount of computation involved In addition, by making the spline used in the solid beckbone model pass directly through the alpha carbon locations, all of the static models developed earlier, such as the wireframe model and the solvent-accessible surface model, can be extensively exploited as part of the animation system presented in this dissertation.

No matter what sources for the determination of folding intermediates are used for the animation, the technique of inbetweening is required to smooth out the motion sequence. Conformational data are intrinsically discrete whether the data are taken from hydrogen exchange, nuclear magnetic resonance, or energy minimization. The spatial pattern from one folding intermediate to another does not necessarily exhibit a gradual transition. Therefore, the introduction of the inbetweening technique is required to fill in gaps in the transition, since the visual illusion leading to the animation cannot be anticipated with such a discrete sequence of conformation. Despite its popularity in the context of computer animation, the inbetweening technique has not been applied in any existing molecular animation system. This dissertation has developed a prototype molecular animation system incorporating the inbetweening technique.

Splines are used in the prototype system in two ways. They are used for the generation of the curve passing through the backbone atoms and they are used for the generation of the motion path of individual atoms over a time span. In implementing these concepts, problems associated with conventional interpolating splines had to be resolved. Those interpolating splines with parametric continuity could not satisfy our needs because of the abnormal behavior called wiggles or loops. If the parametrically-continuous spline is employed, the backbone shape and the motion path cannot be free of wiggles, due to the unduly strained constraints applied to the spline. That is, at a certain data point, both the magnitude and direction of the left and right tangents should be identical. Another class of splines, known as visually continuous splines in CAGD literature, encompasses a wider range of spline shapes while assuring the same smoothness visually as the parametrically-continuous spline.

Although an algorithmic approach to produce this curve was presented previously, no explicit formulation of this class of splines as yet exists. In previous approaches, recursive iteration based upon the construction of a Bézier polygon was mandatory to calculate the curve, and the complexity of this algorithm has limited popular use of this class of splines. This dissertation derives and proves an explicit matrix representation of the visually-continuous interpolating splines. The formulation was derived by allowing the insertion of variable knot parameters, with the same parabolic blending as was used for the formulation of the well-known Overhauser curve. Based on mathematical representation, three characteristic types of the visually-continuous splines were developed.

An important aspect of the formulation of the splines presented in this dissertation is the consistent use of vector notation. Exploiting the advantage of vectors, a simple graphical interface for the generation of a motion path was presented. An animator designing the motion path of an object can simply draw a line to dictate the direction of the motion at corresponding data points. Such an extension of the visually-continuous splines may be applied to any applications involving animation.

Another area of concern is the control of motion speed in the animation. A control strategy was required to ensure the proper motion speed of individual atoms while the sequence of folding intermediates are displayed. Since the splines used in computer graphics are generally made to be parametric for certain purposes, the magnitude of the derivatives on the curve, and thus the inter-frame distances, are parametric as well. Recent research in this area of animation suggested the possibility of controlling the speed using a means of calculation in the continuous domain involving estimation of the integration. However, in our prototype system, the folding intermediates corresponding to key frames must be part of the animation sequence. Our prototype system could not apply the continuous domain approach, since this technique does not necessarily hit the key frames. A discrete domain alternative—the Incremental Knot Spacing method, combined with the Lookahead Adjustment and Averaging Adjustment—presented in this dissertation was shown to produce satisfactory control of motion speed between frames. In addition, it is guaranteed to pass through the keyframe positions, so that the folding intermediates can be treated and restored as part of the animation sequence.

The key contributions of this research, to computer graphics in general and molecular graphics in particular, can be summarized as follows:

1. Analytic mathematical formulation of visually-continuous splines, known as the G^1 class splines.

2. Development of a mathematical tool for the design of motion path in general animation systems, as an extension of the G^1 class splines.

3. Development of a method for the control of the motion speed in a spline-based animation system.

4. Development of a molecular model for the animated display of protein folding.

5. Introduction of the inbetweening technique to the field of molecular animation.

BIBLIOGRAPHY

Aho83 Aho, A. V , Hopcroft, J E., Ullman, J D , *Data Structures and Algorithms*, 1983, Addison-Wesley, Reading, Massachusetts, New York

Aki70 Akima, H , "A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures," Journal of ACM, vol.1, no.4, pp.589-602, 1970

Ath83 Atherton, P R., "A Scan-line Hidden Surface Removal Procedure for Constructive Solid Geometry," Computer Graphics, vol.17, no.3, pp.73-82, 1983

Bae69 Baecker, R., "Interactive Computer-Mediated Animation," MAC-TR-61, Ph.D Thesis, Massachusetts Institute of Technology, 1969

Bars84 Barsky, B A , "Exponential and Polynomial Methods for Applying Tension to an Interpolating Spline Curve," Computer Vision, Graphics, and Image Processing, vol.27, pp 1-18, 1984

Bars83 Barsky, B A , and Beatty, J C., "Local Control of Bias and Tension in Beta-splines," Computer Graphics, vol 17, no.3, pp 193-218, 1983

Bars88 Barsky, B A , and Kunii,T L., *Computer Graphics and Geometric Modeling Using Beta-Splines*, 1988, Springer-Verlag, New York

Bart87 Bartels, R H , Beatty, J C , and Barsky, B A , *An Introduction to Splines For Use In Computer Graphics and Geometric Modeling*, 1983, Morgan Kaufmann Pusblisher, Los Altos, California.

Bart89 Bartels, R H , and Hardtke, L , "Speed Adjustment for Key-Frame Interpolation," Graphics Interface 89, pp 14-19, 1989

Béz70 Bézier, P E , *Numerical Control-Mathematics and Applications*, 1972, John Wiley & Sons, New York.

Béz74 Bézier, P, "Mathematical and Practical Possibilities of UNISURF," Computer Aided Geometric Design, Barnhill and Riesenfeld (eds), 1974, Academic Press, New York.

Bres77 Bresenham, J E., "An Incremental Algorithm for Digital Display of Circular Arcs," Communication of ACM, vol 20, no.2, pp 100-106, 1977

Brew77 Brewere, J A., and Anderson, D C, "Visual Interaction With Overhauser Curves And Surfaces," Computer Graphics, vol 11, no.2, pp 132-137, 1977

Bro83 Brooks, B R, Bruccoleri, R. E, Olafson, B D, States, D J, Swaminathan, S, and Kaplus, M., "CHARMM. A Program for Macromolecular Energy Minimization, and Dynamics Calculation," Journal of Computational Chemistry, vol 4, no 2, pp 187-217, 1983

Bur71 Burtnyk, N, and Wein, M, "Computer-Generated Key-Frame Animation," Journal of SMPTE, vol.80, pp 149-153, 1971

Bur76 Burtnyk, N, and Wein, M, "Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation,", Communication of ACM, vol.19, no 10, pp.564-569, 1976

Car76 Carmo, M. P, *Differential Geometry of Curves and Surfaces*, 1976, Prentice-Hall, Englewood Cliffs, New Jersey

Car86 Carson, M., and Bugg, C E., "Algorithm for Ribbon Models of Proteins," Journal of Molecular Graphics, vol.4, no.2, pp 121-122, 1986.

Cat74 Catmull, E, and Rom, R., "A Class of Local Interpolating Splines," Computer Aided Geometric Design, Barnhill and Riesenfeld (eds), 1974, Academic Press, New York.

Cat78 Catmull, E., "The Problems of Computer-Assisted Animation," Computer Graphics, vol 12, no.3, pp 348-353, 1978

Che88 Cherfils, J., Vaney, M C., Morize, I., Surcouf, E., Colloc'h, N, and Mornon, J P, "MANOSK. A Graphics Program for Analyzing and Modeling Molecular Structure and Functions," Journal of Molecular Graphics, vol 6, pp 155-160, 1988

Cho84 Chothia, C , "Principles that Determine the Structure of Proteins," Annual Review of Biochemistry, vol 52, pp 537-572, 1984

Chu90 Chun P W , Milov, D E , and Jou, W , "Quantitative Reappraisal of General Expressions for Multivalent Protein Binding in Subunit-Exchange Chromatography," Biophysical Chemistry, vol.38, pp 131-141, 1990.

Chu91 Chun, P W , and Jou, W , "Molecular Conformation of Ubiquitinated Structures and the Implications for Regulatory Function,' Journal of Molecular Graphics, in press

Cla89 Clark, M , Cramer III, R. D , and Opdenbosch, N E, "Validation of the General Purpose Tripos 5.2 Force Field," Journal of Computational Chemistry, vol.10, no 8, pp 982-1012, 1989

Con83 Connolly, M L., "Solvent-Accessible Surfaces of Proteins and Nucleic Acids," Science, vol.221, no 4621, pp 709-713, 1983

Coo74 Coons, S A., "Surface Patches and B-Spline Curves,' Computer Aided Geometric Design, Barnhill and Riesenfeld (eds), 1974, Academic Press, New York.

Cro77 Crow, F C , "The Aliasing Problem in Computer Generated Shaded Images," Communication of ACM, vol.20, no 11, pp 799-805, 1977

Cro81 Crow, F C , "A Comparison of Antialiasing Techniques," IEEE Computer Graphics and Applications, vol 1, no 1, pp 40-48, January, 1981

Dat86 Date, C J , *An Introduction to Database Systems Volume 1*, 1986, Addison-Wesley, New York

deB78 deBoor, C , *A Practical Guide to Splines*, 1978, Springer-Verlag, New York.

Del84 Delhaise, P, Bardiaux, M , and Wodak, S., "Interactive Computer Animation of Macromolecules," Journal of Molecular Graphics, vol 1, no.2, pp 103-106, 1984

Der88 Derose, T D , and Barsky, B A., "Geometric Continuity, Shape Parameters, and Geometric Constructions for Catmull-Rom Splines,' ACM Transactions on Graphics, vol.7, no 1, pp 1-41, 1988.

Elm89 Elmasri, R., and Navathe, S B, *Fundamentals of Database Systems*, 1989, Benjamin/Cummings Publishing Company, California.

Far85 Farin, G "Some Remarks on v^2- Splines," Computer Aided Geometric Design, vol.2, no.2, pp 325-328, 1985

Far90 Farin, G, *Curves and Surfaces for Computer Aided Geometric Design*, 1990, Academic Press, New York.

Fel80 Feldmann, R. J., and Levitt, M, "Molecular Dynamics of Bovine Pancreatic Trypsin Inhibitor," Film, 1980 Ref "Interactive Computer Surface Graphics Approach to Study of the Active Site of Bovine Trypsin,", Proceedings of National Academy of Science, vol 75, no.11, pp.5409-5412, 1978

Fle86 Fletterick, R., and Zoller, M., *Current Communications in Molecular Biology*, 1986, Cold Spring Harbor Laboratory, New York

Fol82 Foley, J D., and Van Dam, A, *Fundamentals of Interactive Computer Graphics*, 1982, Addison-Wesley, Reading, Massachusetts, New York.

Fol86 Foley, T A., "Local Control of Interval Tension Using Weighted Splines," Computer Aided Geometric Design, vol.3, pp.281-294, 1986

For72 Forrest, A R., "Interactive Interpolation and Approximation by Bezier Polynomials," Computer Journal, vol 15, no 1, pp 71-79, 1972.

For72 Forrest, A R., "On Coons And Other Methods For The Representation of Curved Surfaces," Computer Graphics and Image Processing, vol 1, pp.341-359, 1972.

Fra81 Franklin, W R., "An Exact Hidden Sphere Algorithm That Operates in Linear Time," Computer Graphics and Image Processing, vol 15, pp.364-379, 1981

Fu87 Fu, K S, Gonzalez, R C, and Lee, C S, *Robotics Control, Sensing, Vision, and Intelligence*, 1987, McGraw-Hill, New York

Fuj86 Fujita, T , Swanson, S and Meyer E , "Intermolecular Enzyme-ligand Animation in the Active Site of Porcine Pancreatic Elastase with Acetyl-alanine-proline-alanine by Means of Molecular Dynamics Calculation," Journal of Molecular Graphics, vol 4, no.4, pp.208-212, 1986

Ghé82 Ghélis, C.,Yon, J., *Protein Folding*, Academic Press, New York.

Gla89 Glassner, A S., "An Overview of Ray Tracing," Ray Tracing, Glassner, A S (eds), 1989, Academic Press, San Diego, California

Gor74a Gordon, W J , and Riesenfeld, R F , "Bernstein-Bezier Methods for the Computer-Aided Design of Free-Form Curves and Surfaces," Journal of ACM, vol 21, no 2, pp 293-310, 1974.

Gor74b Gordon, W J , and Riesenfeld, R. F , "B-Spline Curves and Surfaces," Computer Aided Geometric Design, Barnhill and Riesenfeld (eds), 1974, Academic Press, New York.

Har87 Harrington, S , *Computer Graphics*, 1987, McGraw-Hill, New York

Hea86 Hearn, D , and Baker, M. P , *Computer Graphics*, 1986, Prentice-Hall, Englewood Cliffs, New Jersey

Her78 Herbison-Evans, D , "Nude 2 A Numeric Utility Displaying Ellipsoid Solids, Version2," Computer Graphics, vol 15, no 1, pp 354-356, 1978

Jac71 Jacquez, J , *A First Course In Computing And Numerical Methods*, 1970, Addison-Wesley, Reading, Massachusetts, New York

Jac81 Jacques, I., and Judd, C , *Numerical Analysis*, 1987, Chapman and Hall, New York

Jou91 Jou, W , and Chun, P W , "Molecular Mechanics of the Formation of Cholic Acid Micelles," Journal of Molecular Graphics, in press.

Kim82 Kim, P S , and Baldwin, R. L., "Specific Intermediates in the Folding Reactions of Small Proteins and the Mechanism of Protein Folding," Annual Review of Biochemistry, vol 51, pp.459-489, 1982.

Kno81 Knowlton, K., "Computer-Aided Definition, Manipulation, and Depiction of Objects Composed of Spheres," Computer Graphics, vol.15, no 1, pp 48-71, 1981

Koc84 Kochanek, D H U, and Bartels, R. H., "Interpolating Splines with Local Tension, Continuity and Bias Control," Computer Graphics, vol.18, no 3, pp 33-41, 1984

Lan80 Lane, J M, Carpenter, L. C., Whitted, J T, and Blinn, J F, "Scan Line Methods Displaying Parametrically Defined Surfaces," Communications of ACM, vol 23, no 1, pp.23-34, 1980

Lan86 Lancaster, P, and Salkauskas, K., *Curve and Surface Fitting*, 1986, Academic Press, London

Las87 Lasseter, J, "Principles of Traditional Animation Applied to 3D Computer Animation," Computer Graphics, vol 21, no 4, pp 35-44, 1987

Lev66 Levinthal, C, "Molecular Modelbuilding by Computer," Scientific American, vol.214, no.6, pp 42-52, 1966

Lev83a Levitt, M "Protein Folding by Restrained Energy Minimization and Molecular Dynamics," Journal of Molecular Biology, vol 170, pp.723-764, 1983

Lev83b Levitt, M, "Molecular Dynamics of Native Protein 1 Computer Simulation of Trajectories," Journal of Molecular Biology, vol 168, pp.595-620, 1983

Lou83 Loule, A, and Somorjai, R, "Differential Geometry of Proteins," Journal of Molecular Biology, vol 168, pp 143-162, 1983

Man74 Manning, J R., 'Continuity Conditions For Spline Curves," The Computer Journal, vol.17, no.2, pp 181-186, 1974.

Mas86 Mastin, C W, "Parameterization In Grid Generation," Computer Aided Design, vol.18, no 1, pp.22-24, 1986

Max78 Max, N L., "ATOMLLL-Atoms with Shading and Highlights," Computer Graphics, vol 12, no 3, pp 165-173, 1978

Max83 Max, N L., "Computer Representation of Molecular Surfaces," IEEE Computer Graphics and Applications, vol.3, pp 21-29, August, 1983

Max88 Max, N L , and Getzoff, E D , 'Spherical Harmonic Molecular Surfaces,' IEEE Computer Graphics and Applications vol 8, no.4, pp 42-50, July, 1988

Met77 Metzler, D , *Biochemistry*, 1977, Academic Press, New York

Mil91 Milov, D E , Jou, W , Shireman, R B , and Chun, P W , 'The Effect of Bile Salt on Carbonic Anhydrase,' Hepatology, in press.

New79 Newman, W M , Sproull, R F , *Principles of Interactive Computer Graphics*, 1979, McGraw Hill, New York.

Nie74 Nielson, G M , 'Some Piecewise Polynomial Alternatives to Splines Under Tension,' Computer Aided Geometric Design, Barnhill and Riesenfeld (eds), 1974, Academic Press, New York.

Ove68 Overhauser, A W , *Analytic Definition of Curves and Surfaces by Parabolic Blending*, 1968, Scientific Research Staff Publication, Ford Motor Company

Pal88 Palmer, T C , and Hausheer, F H., "Context free Spheres. A New Method for Rapid CPK Image Generation," Journal of Molecular Graphics, vol.6, pp 149-154, 1988

Piq86 Pique, M E , "Technical Trends in Molecular Graphics," in *Current Communications in Molecular Biology*, 1986, Cold Spring Harbor Laboratory, New York

Por78 Porter, T K , 'Spherical Shading,' Computer Graphics, vol 12, no.3, pp.282-285, 1978

Pre88 Press, W H , Flannery, B P , Teukolsky, S S , and Vetterling, W T , *Numerical Recipes in C*, 1988. Cambridge University Press, New York

Ree81 Reeves, W., "Inbetweening for Computer Animation Utilizing Moving Point Constraints," Computer Graphics, vol 15, no.3, pp 263-269, 1981

Ric81 Richardson, J S , 'The Anatomy and Taxanomy of Protein Structure,' Advanced Protein Chemistry, vol 34, pp 167 320, 1981

Ric91 Richards, F M , "The Protein Folding Problem," Scientific American, pp.54-63, Jan 1991

Rog76 Rogers, D F , and Adams, J A , *Mathematical Elements for Computer Graphics*, 1976 McGraw-Hill New York.

Ros81 Rossmann, J G and Argos, P , "Protein Folding," Annual Review of Biochemistry vol.50, pp 497-532, 1981

Sal84 Salkausas, K., "C^1 Splines for Interpolation of Rapidly Varying Data," Rocky Mountain Journal of Mathematics, vol 14, pp.239-250, 1984

Sam90 Samet, H , *The Design and Analysis of Spatial Data Structures*, 1990, Addison-Wesley, Reading, Massachusetts, New York.

Sch66 Schweikert, D G , "An Interpolation Curve Using a Spline in Tension," Journal of Math. Phys , vol.45, pp.312-317, 1966

Sch79 Schulz, R. H., Schirmer, R. H , *Principles of Protein Structure*, 1979, Springer Verlag, New York.

Smi83 Smith, A. R., "Spline Tutorial Notes," ACM SIGGRAPH 88 Course Notes Introduction to Computer Animation, pp 131-142, 1988

Sta78 Staudhammer, J , "On Display of Space Filling Atomic Models In Real-Time," Computer Graphics, vol.12, no.3, pp 167-172, 1978

Sta86 Staudhammer, J , Khurana, A S , "Display of Moleculer Models with Interactive Graphics," IEEE Computer Graphics and Applications, vol.6, no.1, pp.26-31, Jan., 1986

Sta88 Staudhammer, J., Livadas, P E , Wang, S L., and Zhou, X , "Molecular Structure Visualization and Optimal Surface Reconstruction," NCGA Conference Proceedings, pp 168-177, March, 1988

Ste85 Steketee, S N , and Badler, N , "Parametric Keyframe Interpolation Incorporating Kinetic Adjustmant and Phrasing Control," Computer Graphics, vol 19, no 3, pp.255-262, 1985

Str81 Stryer,L., *Biochemistry*, 1981, W.H. Freeman And Company, New York.

Tha85 Thalmann, N M , and Thalmann, D , *Computer Animation Theory and Practice*, 1985, Springer-Verlag, Tokyo

Tha88 Thalmann, N. M., and Thalmann, D., *New Trends in Computer Graphics: Proceedings of Computer Graphics International '88*, 1988, Springer-Verlag, New York.

Tho81 Thomas, F., and Johnsonton, O., *Disney Animation: The Illusion of Life*, 1981, Abbeville Press, New York.

Tod83 Todd, S., and Gillett, J., "Animation in the Winchester Graphic System," Journel of Molecular Graphics, vol.1, no.2, pp.38-42, 1983.

Syb89 "SYBYL," Molecular Modeling Software, *Theory Manual*, Tripos Associates, a Subsidiary of Evans and Sutherland, St. Louis, Missouri, 1989.

Voe90 Voet, D., and Voet, J. G., *Biochemistry*, 1990, John Wiley & Sons, New York.

Wan89 Wang, S. C., "On Visibility Determination in Surfaces with Partially Ordered Data," Ph.D Thesis, Univ. of Florida, 1989.

Whi80 Whitted, J. T., "An Improved Illumination Model for Shaded Display," Communication of ACM, vol.23, no.6, pp.343-349, 1980.

Wri88 Wright, P. E., Dyson, H. J., and Lerner, R. A., "Conformation of Peptide Fragments of Proteins in Aqueous Solutions: Implications for Initiation of Protein Folding," Biochemistry, vol.27, no.19, pp.7167-7175, 1988.

Wri73 Wright, T. J., 'A Two-Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables.' IEEE Transactions on Computer, vol.C-22, pp.28-33, 1973.

BIOGRAPHICAL SKETCH

Wouseok Jou was born in Pohang, Korea, on November 7, 1955 He received his B.S degree in electronic engineering from Seoul National University, Seoul, Korea, in 1983 After graduation, he was employed as a system engineer by International Business Machines, incorporated, Seoul, Korea. His duties included instruction on IMS database, SNA concepts, and System/370 Assembler Language.

He was admitted to the master's program in electrical engineering at the University of Florida in the fall semester of 1985 and worked as a research assistant at the Database Research Center, Department of Computer and Information Science He received his M S degree in electrical engineering in the fall semester of 1987 With the advice of Dr John Staudhammer and Dr Paul W Chun, he continued in a Ph.D program at the University of Florida He passed the doctoral qualifying examination in the fall semester, 1989.

His current research interests are computer graphics image processing, database design, digital signal processing, and VLSI design areas

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

John Staudhammer

John Staudhammer, Chairman
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Paul W. Chun
Professor of Biochemistry and Molecular Biology

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jack R. Smith
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Jose C. Principe
Associate Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality as a dissertation for the degree of Doctor of Philosophy

Chang W. Park

Chang W Park
Assistant Professor of Chemical Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy

August 1991

Herbert G. Bevis

for Winfred M Phillips
Dean, College of Engineering

Madelyn M. Lockhart
Dean, Graduate School